Package 'gtsummary'

August 11, 2020

Title Presentation-Ready Data Summary and Analytic Result Tables

Version 1.3.3

Description Creates presentation-ready tables summarizing data sets, regression models, and more. The code to create the tables is concise and highly customizable. Data frames can be summarized with any function, e.g. mean(), median(), even user-written functions. Regression models are summarized and include the reference rows for categorical variables. Common regression models, such as logistic regression and Cox proportional hazards regression, are automatically identified and the tables are pre-filled with appropriate column headers.

License MIT + file LICENSE

URL https://github.com/ddsjoberg/gtsummary, http://www.danieldsjoberg.com/gtsummary/

BugReports https://github.com/ddsjoberg/gtsummary/issues

Depends R (>= 3.4)

Imports broom (>= 0.7.0),
broom.mixed (>= 0.2.6),
dplyr (>= 1.0.1),
forcats (>= 0.5.0),
glue (>= 1.4.1),
gt (>= 0.2.2),
knitr (>= 1.29),
lifecycle (>= 0.2.0),
magrittr (>= 1.5),
purrr (>= 0.3.4),
rlang (>= 0.4.7),
stringr (>= 1.4.0),
tibble (>= 3.0.3),
tidyr (>= 1.1.1),
tidyselect (>= 1.1.0),
usethis (>= 1.6.1)

Suggests car,
covr,
flextable (>= 0.5.10),
geepack,

Hmisc,
huxtable (>= 5.0.0),
kableExtra,
lme4,
officer,
pkgdown,
rmarkdown,
scales,
spelling,
survey,
survival,
testthat

VignetteBuilder knitr

RdMacros lifecycle

Encoding UTF-8

Language en-US

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.1

R topics documented:

`add_global_p` *Adds the global p-value for a categorical variables*

Description

This function uses car::Anova with argument `type = "III"` to calculate global p-values for categorical variables. Output from `tbl_regression` and `tbl_uvregression` objects supported.

Usage

```
add_global_p(x, ...)
```

Arguments

`x`	`tbl_regression` or `tbl_uvregression` object
`...`	Further arguments passed to or from other methods.

Note

If a needed class of model is not supported by car::Anova, please create a GitHub Issue to request support.

Author(s)

Daniel D. Sjoberg

See Also

add_global_p.tbl_regression, add_global_p.tbl_uvregression

---

add_global_p.tbl_regression *Adds the global p-value for categorical variables*

Description

This function uses car::Anova with argument type = "III" to calculate global p-values for categorical variables.

Usage

```
## S3 method for class 'tbl_regression'
add_global_p(
  x,
  include = x$table_body$variable[x$table_body$var_type %in% c("categorical",
    "interaction")],
  type = NULL,
  keep = FALSE,
  quiet = NULL,
  ...,
  terms = NULL
)
```

Arguments

x	Object with class tbl_regression from the tbl_regression function
include	Variables to calculate global p-value for. Input may be a vector of quoted or unquoted variable names. tidyselect and gtsummary select helper functions are also accepted. Default is NULL, which adds global p-values for all categorical and interaction terms.
type	Type argument passed to car::Anova. Default is "III"
keep	Logical argument indicating whether to also retain the individual p-values in the table output for each level of the categorical variable. Default is FALSE
quiet	Logical indicating whether to print messages in console. Default is FALSE
...	Additional arguments to be passed to car::Anova
terms	DEPRECATED. Use include= argument instead.

Value

A tbl_regression object

Note

If a needed class of model is not supported by car::Anova, please create a GitHub Issue to request support.

Example Output

Author(s)

Daniel D. Sjoberg

See Also

Other tbl_regression tools: `add_nevent.tbl_regression()`, `add_q()`, `bold_italicize_labels_levels`, `combine_terms()`, `inline_text.tbl_regression()`, `modify`, `tbl_merge()`, `tbl_regression()`, `tbl_stack()`

Examples

```
tbl_lm_global_ex1 <-
  lm(marker ~ age + grade, trial) %>%
  tbl_regression() %>%
  add_global_p()
```

---

add_global_p.tbl_uvregression

Adds the global p-value for categorical variables

Description

This function uses car::Anova with argument `type = "III"` to calculate global p-values for categorical variables.

Usage

```
## S3 method for class 'tbl_uvregression'
add_global_p(
  x,
  type = NULL,
  include = everything(),
  keep = FALSE,
  quiet = NULL,
  ...
)
```

Arguments

x	Object with class tbl_uvregression from the tbl_uvregression function
type	Type argument passed to car::Anova. Default is "III"
include	Variables to calculate global p-value for. Input may be a vector of quoted or unquoted variable names. tidyselect and gtsummary select helper functions are also accepted. Default is everything().
keep	Logical argument indicating whether to also retain the individual p-values in the table output for each level of the categorical variable. Default is FALSE
quiet	Logical indicating whether to print messages in console. Default is FALSE
...	Additional arguments to be passed to car::Anova.

Value

A tbl_uvregression object

Example Output

Author(s)

Daniel D. Sjoberg

See Also

Other tbl_uvregression tools: add_nevent.tbl_uvregression(), add_q(), bold_italicize_labels_levels, inline_text.tbl_uvregression(), modify, tbl_merge(), tbl_stack(), tbl_uvregression()

Examples

```
tbl_uv_global_ex2 <-
  trial[c("response", "trt", "age", "grade")] %>%
  tbl_uvregression(
    method = glm,
    y = response,
    method.args = list(family = binomial),
    exponentiate = TRUE
  ) %>%
  add_global_p()
```

---

add_n *Adds column with N to gtsummary table*

Description

Adds column with N to gtsummary table

Usage

```
add_n(x, ...)
```

Arguments

x	Object created from a gtsummary function
...	Additional arguments passed to other methods.

Author(s)

Daniel D. Sjoberg

See Also

add_n.tbl_summary, add_n.tbl_svysummary, add_n.tbl_survfit

---

add_n.tbl_summary *Add column with N*

Description

For each variable in a tbl_summary table, the add_n function adds a column with the total number of non-missing (or missing) observations

Usage

```
## S3 method for class 'tbl_summary'
add_n(
  x,
  statistic = "{n}",
  col_label = "**N**",
  footnote = FALSE,
  last = FALSE,
  missing = NULL,
  ...
)

## S3 method for class 'tbl_svysummary'
add_n(
  x,
  statistic = "{n}",
  col_label = "**N**",
  footnote = FALSE,
  last = FALSE,
  missing = NULL,
  ...
)
```

Arguments

x	Object with class tbl_summary from the tbl_summary function or with class tbl_svysummary from the tbl_svysummary function
statistic	String indicating the statistic to report. Default is the number of non-missing observation for each variable, statistic = "{n}". Other statistics available to report include:

- `"{N}"` total number of observations,
- `"{n}"` number of non-missing observations,
- `"{n_miss}"` number of missing observations,
- `"{p}"` percent non-missing data,
- `"{p_miss}"` percent missing data The argument uses glue::glue syntax and multiple statistics may be reported, e.g. `statistic = "{n} / {N} ({p}%)"`

`col_label` String indicating the column label. Default is `"**N**"`

`footnote` Logical argument indicating whether to print a footnote clarifying the statistics presented. Default is `FALSE`

`last` Logical indicator to include N column last in table. Default is `FALSE`, which will display N column first.

`missing` DEPRECATED. Logical argument indicating whether to print N (`missing = FALSE`), or N missing (`missing = TRUE`). Default is `FALSE`

`...` Not used

Value

A `tbl_summary` or `tbl_svysummary` object

Example Output

Author(s)

Daniel D. Sjoberg

See Also

Other tbl_summary tools: `add_overall()`, `add_p.tbl_summary()`, `add_q()`, `add_stat_label()`, `bold_italicize_labels_levels`, `inline_text.tbl_summary()`, `inline_text.tbl_survfit()`, `modify`, `tbl_merge()`, `tbl_stack()`, `tbl_summary()`

Other tbl_svysummary tools: `add_overall()`, `add_p.tbl_svysummary()`, `add_q()`, `add_stat_label()`, `modify`, `tbl_merge()`, `tbl_stack()`, `tbl_svysummary()`

Examples

```
tbl_n_ex <-
  trial[c("trt", "age", "grade", "response")] %>%
  tbl_summary(by = trt) %>%
  add_n()
```

add_n.tbl_survfit *Add column with number of observations*

Description

Experimental For each `survfit()` object summarized with `tbl_survfit()` this function will add the total number of observations in a new column.

Usage

```
## S3 method for class 'tbl_survfit'
add_n(x, ...)
```

Arguments

`x`	object of class "`tbl_survfit`"
`...`	Not used

Example Output

See Also

Other tbl_survfit tools: `add_nevent.tbl_survfit()`, `add_p.tbl_survfit()`, `modify`, `tbl_merge()`, `tbl_stack()`, `tbl_survfit()`

Examples

```
library(survival)
fit1 <- survfit(Surv(ttdeath, death) ~ 1, trial)
fit2 <- survfit(Surv(ttdeath, death) ~ trt, trial)

# Example 1 ----------------------------------
add_n.tbl_survfit_ex1 <-
  list(fit1, fit2) %>%
  tbl_survfit(times = c(12, 24)) %>%
  add_n()
```

add_nevent *Add number of events to a regression table*

Description

Adds a column of the number of events to tables created with tbl_regression or tbl_uvregression. Supported model types include GLMs with binomial distribution family (e.g. stats::glm, lme4::glmer, and geepack::geeglm) and Cox Proportion Hazards regression models (survival::coxph).

Usage

```
add_nevent(x, ...)
```

Arguments

x	tbl_regression or tbl_uvregression object
...	Additional arguments passed to or from other methods.

Author(s)

Daniel D. Sjoberg

See Also

add_nevent.tbl_regression, add_nevent.tbl_uvregression, tbl_regression, tbl_uvregression

---

add_nevent.tbl_regression

Add number of events to a regression table

Description

This function adds a column of the number of events to tables created with tbl_regression. Supported model types include GLMs with binomial distribution family (e.g. stats::glm, lme4::glmer, and geepack::geeglm) and Cox Proportion Hazards regression models (survival::coxph).

The number of events is added to the internal `.$table_body` tibble, and not printed in the default output table (similar to N). The number of events is accessible via the inline_text function for printing in a report.

Usage

```
## S3 method for class 'tbl_regression'
add_nevent(x, quiet = NULL, ...)
```

Arguments

x	tbl_regression object
quiet	Logical indicating whether to print messages in console. Default is FALSE
...	Not used

Value

A tbl_regression object

Example Output

Author(s)

Daniel D. Sjoberg

See Also

Other tbl_regression tools: add_global_p.tbl_regression(), add_q(), bold_italicize_labels_levels, combine_terms(), inline_text.tbl_regression(), modify, tbl_merge(), tbl_regression(), tbl_stack()

Examples

```
add_nevent_ex <-
  glm(response ~ trt, trial, family = binomial) %>%
  tbl_regression() %>%
  add_nevent()
```

---

add_nevent.tbl_survfit

Add column with number of observed events

Description

Experimental For each survfit() object summarized with tbl_survfit() this function will add the total number of events observed in a new column.

Usage

```
## S3 method for class 'tbl_survfit'
add_nevent(x, ...)
```

Arguments

x	object of class 'tbl_survfit'
...	Not used

Example Output

See Also

Other tbl_survfit tools: add_n.tbl_survfit(), add_p.tbl_survfit(), modify, tbl_merge(), tbl_stack(), tbl_survfit()

Examples

```
library(survival)
fit1 <- survfit(Surv(ttdeath, death) ~ 1, trial)
fit2 <- survfit(Surv(ttdeath, death) ~ trt, trial)

# Example 1 ----------------------------------
add_nevent.tbl_survfit_ex1 <-
  list(fit1, fit2) %>%
  tbl_survfit(times = c(12, 24)) %>%
  add_n() %>%
  add_nevent()
```

`add_nevent.tbl_uvregression` *Add number of events to a regression table*

Description

Adds a column of the number of events to tables created with tbl_uvregression. Supported model types include GLMs with binomial distribution family (e.g. stats::glm, lme4::glmer, and geepack::geeglm) and Cox Proportion Hazards regression models (survival::coxph).

Usage

```
## S3 method for class 'tbl_uvregression'
add_nevent(x, ...)
```

Arguments

`x`	`tbl_uvregerssion` object
`...`	Not used

Value

A `tbl_uvregression` object

Reporting Event N

The number of events is added to the internal `.$table_body` tibble, and printed to the right of the N column. The number of events is also accessible via the inline_text function for printing in a report.

Example Output

Author(s)

Daniel D. Sjoberg

See Also

Other tbl_uvregression tools: `add_global_p.tbl_uvregression()`, `add_q()`, `bold_italicize_labels_levels`, `inline_text.tbl_uvregression()`, `modify`, `tbl_merge()`, `tbl_stack()`, `tbl_uvregression()`

Examples

```
tbl_uv_nevent_ex <-
  trial[c("response", "trt", "age", "grade")] %>%
  tbl_uvregression(
    method = glm,
    y = response,
    method.args = list(family = binomial)
  ) %>%
  add_nevent()
```

add_overall *Add column with overall summary statistics*

Description

Adds a column with overall summary statistics to tables created by `tbl_summary` or `tbl_svysummary`.

Usage

```
add_overall(x, last, col_label)

## S3 method for class 'tbl_summary'
add_overall(x, last = FALSE, col_label = NULL)

## S3 method for class 'tbl_svysummary'
add_overall(x, last = FALSE, col_label = NULL)
```

Arguments

`x`	Object with class `tbl_summary` from the tbl_summary function or object with class `tbl_svysummary` from the tbl_svysummary function.
`last`	Logical indicator to display overall column last in table. Default is `FALSE`, which will display overall column first.
`col_label`	String indicating the column label. Default is `"**Overall**, N = {N}"`

Value

A `tbl_summary` object or a `tbl_svysummary` object

Example Output

Author(s)

Daniel D. Sjoberg

See Also

Other tbl_summary tools: `add_n.tbl_summary()`, `add_p.tbl_summary()`, `add_q()`, `add_stat_label()`, `bold_italicize_labels_levels`, `inline_text.tbl_summary()`, `inline_text.tbl_survfit()`, `modify`, `tbl_merge()`, `tbl_stack()`, `tbl_summary()`

Other tbl_svysummary tools: `add_n.tbl_summary()`, `add_p.tbl_svysummary()`, `add_q()`, `add_stat_label()`, `modify`, `tbl_merge()`, `tbl_stack()`, `tbl_svysummary()`

Examples

```
tbl_overall_ex <-
  trial[c("age", "grade", "trt")] %>%
  tbl_summary(by = trt) %>%
  add_overall()
```

add_p *Adds p-values to gtsummary table*

Description

Adds p-values to gtsummary table

Usage

```
add_p(x, ...)
```

Arguments

x	Object created from a gtsummary function
...	Additional arguments passed to other methods.

Author(s)

Daniel D. Sjoberg

See Also

add_p.tbl_summary, add_p.tbl_cross, add_p.tbl_svysummary, add_p.tbl_survfit

add_p.tbl_cross *Adds p-value to crosstab table*

Description

Experimental Calculate and add a p-value comparing the two variables in the cross table. Missing values are included in p-value calculations.

Usage

```
## S3 method for class 'tbl_cross'
add_p(x, test = NULL, pvalue_fun = NULL, source_note = NULL, ...)
```

Arguments

x	Object with class `tbl_cross` from the tbl_cross function
test	A string specifying statistical test to perform. Default is "`chisq.test`" when expected cell counts >=5 and "`fisher.test`" when expected cell counts <5.
pvalue_fun	Function to round and format p-value. Default is style_pvalue, except when `source_note = TRUE` when the default is `style_pvalue(x, prepend_p = TRUE)`
source_note	Logical value indicating whether to show p-value in the {gt} table source notes rather than a column.
...	Not used

Example Output

Author(s)

Karissa Whiting

See Also

Other tbl_cross tools: `inline_text.tbl_cross()`, `tbl_cross()`

Examples

```
# Example 1 ----------------------------------
add_p_cross_ex1 <-
  trial %>%
  tbl_cross(row = stage, col = trt) %>%
  add_p()

# Example 2 ----------------------------------
add_p_cross_ex2 <-
  trial %>%
  tbl_cross(row = stage, col = trt) %>%
  add_p(source_note = TRUE)
```

---

add_p.tbl_summary *Adds p-values to summary tables*

Description

Adds p-values to tables created by `tbl_summary` by comparing values across groups.

Usage

```
## S3 method for class 'tbl_summary'
add_p(
  x,
  test = NULL,
  pvalue_fun = NULL,
  group = NULL,
  include = everything(),
  exclude = NULL,
  ...
)
```

Arguments

`x`	Object with class `tbl_summary` from the tbl_summary function
`test`	List of formulas specifying statistical tests to perform, e.g. `list(all_continuous() ~ "t.test", all_categorical() ~ "fisher.test")`. Options include • `"t.test"` for a t-test,

- "aov" for a one-way ANOVA test,
- "wilcox.test" for a Wilcoxon rank-sum test,
- "kruskal.test" for a Kruskal-Wallis rank-sum test,
- "chisq.test" for a chi-squared test of independence,
- "chisq.test.no.correct" for a chi-squared test of independence without continuity correction,
- "fisher.test" for a Fisher's exact test,
- "lme4" for a random intercept logistic regression model to account for clustered data, lme4::glmer(by ~ variable + (1 | group),family = binomial). The by argument must be binary for this option.

Tests default to "kruskal.test" for continuous variables, "chisq.test" for categorical variables with all expected cell counts >=5, and "fisher.test" for categorical variables with any expected cell count <5. A custom test function can be added for all or some variables. See below for an example.

pvalue_fun — Function to round and format p-values. Default is style_pvalue. The function must have a numeric vector input (the numeric, exact p-value), and return a string that is the rounded/formatted p-value (e.g. pvalue_fun = function(x) style_pvalue(x,digits = 2) or equivalently, purrr::partial(style_pvalue,digits = 2)).

group — Column name (unquoted or quoted) of an ID or grouping variable. The column can be used to calculate p-values with correlated data (e.g. when the test argument is "lme4"). Default is NULL. If specified, the row associated with this variable is omitted from the summary table.

include — Variables to include in output. Input may be a vector of quoted variable names, unquoted variable names, or tidyselect select helper functions. Default is everything().

exclude — DEPRECATED

... — Not used

Value

A tbl_summary object

Setting Defaults

If you like to consistently use a different function to format p-values or estimates, you can set options in the script or in the user- or project-level start-up file, '.Rprofile'. The default confidence level can also be set. Please note the default option for the estimate is the same as it is for tbl_regression().

- options(gtsummary.pvalue_fun = new_function)

Example Output

Author(s)

Emily C. Zabor, Daniel D. Sjoberg

See Also

See tbl_summary vignette for detailed examples

Other tbl_summary tools: add_n.tbl_summary(), add_overall(), add_q(), add_stat_label(), bold_italicize_labels_levels, inline_text.tbl_summary(), inline_text.tbl_survfit(), modify, tbl_merge(), tbl_stack(), tbl_summary()

Examples

```
# Example 1 ----------------------------------
add_p_ex1 <-
  trial[c("age", "grade", "trt")] %>%
  tbl_summary(by = trt) %>%
  add_p()

# Example 2 ----------------------------------
# Conduct a custom McNemar test for response,
# Function must return a named list of the p-value and the
# test name: list(p = 0.123, test = "McNemar's test")
# The '...' must be included as input
# This feature is experimental, and the API may change in the future
my_mcnemar <- function(data, variable, by, ...) {
  result <- list()
  result$p <- stats::mcnemar.test(data[[variable]], data[[by]])$p.value
  result$test <- "McNemar's test"
  result
}

add_p_ex2 <-
  trial[c("response", "trt")] %>%
  tbl_summary(by = trt) %>%
  add_p(test = response ~ "my_mcnemar")
```

---

add_p.tbl_survfit *Adds p-value to survfit table*

Description

Experimental Calculate and add a p-value

Usage

```
## S3 method for class 'tbl_survfit'
add_p(
  x,
  test = "logrank",
  test.args = NULL,
  pvalue_fun = style_pvalue,
  include = everything(),
  quiet = NULL,
  ...
)
```

Arguments

x	Object of class "tbl_survfit"
test	string indicating test to use. Must be one of "logrank", "survdiff", "petopeto_gehanwilcoxon", "coxph_lrt", "coxph_wald", "coxph_score". See details below
test.args	Named list of additional arguments passed to method in test=. Does not apply to all test types.
pvalue_fun	Function to round and format p-values. Default is style_pvalue. The function must have a numeric vector input (the numeric, exact p-value), and return a string that is the rounded/formatted p-value (e.g. pvalue_fun = function(x) style_pvalue(x,digits = 2) or equivalently, purrr::partial(style_pvalue,digits = 2)).
include	Variables to include in output. Input may be a vector of quoted variable names, unquoted variable names, or tidyselect select helper functions. Default is everything().
quiet	Logical indicating whether to print messages in console. Default is FALSE
...	Not used

test argument

The most common way to specify test= is by using a single string indicating the test name. However, if you need to specify different tests within the same table, the input in flexible using the list notation common throughout the gtsummary package. For example, the following code would call the log-rank test, and a second test of the *G-rho* family.

```
... %>%
  add_p(test = list(trt ~ "logrank", grade ~ "survdiff"),
        test.args = grade ~ list(rho = 0.5))
```

Example Output

See Also

Other tbl_survfit tools: add_n.tbl_survfit(), add_nevent.tbl_survfit(), modify, tbl_merge(), tbl_stack(), tbl_survfit()

Examples

```
library(survival)

gts_survfit <-
  list(survfit(Surv(ttdeath, death) ~ grade, trial),
       survfit(Surv(ttdeath, death) ~ trt, trial)) %>%
  tbl_survfit(times = c(12, 24))

# Example 1 ----------------------------------
add_p_tbl_survfit_ex1 <-
  gts_survfit %>%
  add_p()

# Example 2 ----------------------------------
# Pass `rho=` argument to `survdiff()`
```

```
add_p_tbl_survfit_ex2 <-
  gts_survfit %>%
  add_p(test = "survdiff", test.args = list(rho = 0.5))
```

`add_p.tbl_svysummary` *Adds p-values to svysummary tables*

Description

Adds p-values to tables created by `tbl_svysummary` by comparing values across groups.

Usage

```
## S3 method for class 'tbl_svysummary'
add_p(x, test = NULL, pvalue_fun = NULL, include = everything(), ...)
```

Arguments

`x`	Object with class `tbl_svysummary` from the tbl_svysummary function
`test`	List of formulas specifying statistical tests to perform, e.g. `list(all_continuous() ~ "svy.t.test",all_categorical() ~ "svy.wald.test")`. Options include • `"svy.t.test"` for a t-test adapted to complex survey samples (cf. survey::svyttest), • `"svy.wilcox.test"` for a Wilcoxon rank-sum test for complex survey samples (cf. survey::svyranktest), • `"svy.kruskal.test"` for a Kruskal-Wallis rank-sum test for complex survey samples (cf. survey::svyranktest), • `"svy.vanderwaerden.test"` for a van der Waerden's normal-scores test for complex survey samples (cf. survey::svyranktest), • `"svy.median.test"` for a Mood's test for the median for complex survey samples (cf. survey::svyranktest), • `"svy.chisq.test"` for a Chi-squared test with Rao & Scott's second-order correction (cf. survey::svychisq), • `"svy.adj.chisq.test"` for a Chi-squared test adjusted by a design effect estimate (cf. survey::svychisq), • `"svy.wald.test"` for a Wald test of independence for complex survey samples (cf. survey::svychisq), • `"svy.adj.wald.test"` for an adjusted Wald test of independence for complex survey samples (cf. survey::svychisq), • `"svy.lincom.test"` for a test of independence using the exact asymptotic distribution for complex survey samples (cf. survey::svychisq), • `"svy.saddlepoint.test"` for a test of independence using a saddlepoint approximation for complex survey samples (cf. survey::svychisq), Tests default to `"svy.wilcox.test"` for continuous variables and `"svy.chisq.test"` for categorical variables.
`pvalue_fun`	Function to round and format p-values. Default is style_pvalue. The function must have a numeric vector input (the numeric, exact p-value), and return a string that is the rounded/formatted p-value (e.g. `pvalue_fun = function(x) style_pvalue(x,digits = 2)` or equivalently, `purrr::partial(style_pvalue,digits = 2)`).

include — Variables to include in output. Input may be a vector of quoted variable names, unquoted variable names, or tidyselect select helper functions. Default is `everything()`.

... — Not used

Value

A `tbl_svysummary` object

Example Output

Author(s)

Joseph Larmarange

See Also

Other tbl_svysummary tools: `add_n.tbl_summary()`, `add_overall()`, `add_q()`, `add_stat_label()`, `modify`, `tbl_merge()`, `tbl_stack()`, `tbl_svysummary()`

Examples

```
# Example 1 ----------------------------------
# A simple weighted dataset
add_p_svysummary_ex1 <-
  survey::svydesign(~1, data = as.data.frame(Titanic), weights = ~Freq) %>%
  tbl_svysummary(by = Survived) %>%
  add_p()

# A dataset with a complex design
data(api, package = "survey")
d_clust <- survey::svydesign(id = ~dnum, weights = ~pw, data = apiclus1, fpc = ~fpc)

# Example 2 ----------------------------------
add_p_svysummary_ex2 <-
  tbl_svysummary(d_clust, by = both, include = c(cname, api00, api99, both)) %>%
  add_p()

# Example 3 ----------------------------------
# change tests to svy t-test and Wald test
add_p_svysummary_ex3 <-
  tbl_svysummary(d_clust, by = both, include = c(cname, api00, api99, both)) %>%
  add_p(
    test = list(all_continuous() ~ "svy.t.test",
                all_categorical() ~ "svy.wald.test")
  )
```

add_q *Add a column of q-values to account for multiple comparisons*

Description

Adjustments to p-values are performed with stats::p.adjust.

Usage

```
add_q(x, method = "fdr", pvalue_fun = NULL, quiet = NULL)
```

Arguments

x	a gtsummary object
method	String indicating method to be used for p-value adjustment. Methods from stats::p.adjust are accepted. Default is method = "fdr".
pvalue_fun	Function to round and format p-values. Default is style_pvalue. The function must have a numeric vector input (the numeric, exact p-value), and return a string that is the rounded/formatted p-value (e.g. pvalue_fun = function(x) style_pvalue(x,digits = 2) or equivalently, purrr::partial(style_pvalue,digits = 2)).
quiet	Logical indicating whether to print messages in console. Default is FALSE

Example Output

Author(s)

Esther Drill, Daniel D. Sjoberg

See Also

Other tbl_summary tools: add_n.tbl_summary(), add_overall(), add_p.tbl_summary(), add_stat_label(), bold_italicize_labels_levels, inline_text.tbl_summary(), inline_text.tbl_survfit(), modify, tbl_merge(), tbl_stack(), tbl_summary()

Other tbl_svysummary tools: add_n.tbl_summary(), add_overall(), add_p.tbl_svysummary(), add_stat_label(), modify, tbl_merge(), tbl_stack(), tbl_svysummary()

Other tbl_regression tools: add_global_p.tbl_regression(), add_nevent.tbl_regression(), bold_italicize_labels_levels, combine_terms(), inline_text.tbl_regression(), modify, tbl_merge(), tbl_regression(), tbl_stack()

Other tbl_uvregression tools: add_global_p.tbl_uvregression(), add_nevent.tbl_uvregression(), bold_italicize_labels_levels, inline_text.tbl_uvregression(), modify, tbl_merge(), tbl_stack(), tbl_uvregression()

Examples

```
# Example 1 ----------------------------------
add_q_ex1 <-
  trial[c("trt", "age", "grade", "response")] %>%
  tbl_summary(by = trt) %>%
  add_p() %>%
  add_q()

# Example 2 ----------------------------------
add_q_ex2 <-
  trial[c("trt", "age", "grade", "response")] %>%
  tbl_uvregression(
    y = response,
    method = glm,
    method.args = list(family = binomial),
    exponentiate = TRUE
  ) %>%
  add_global_p() %>%
  add_q()
```

---

add_stat *Add a custom statistic column*

Description

Experimental The function allows a user to add a new column with a custom, user-defined statistic.

Usage

```
add_stat(
  x,
  fns,
  fmt_fun = NULL,
  header = "**Statistic**",
  footnote = NULL,
  new_col_name = NULL
)
```

Arguments

x	tbl_summary object
fns	list of formulas indicating the functions that create the statistic
fmt_fun	for numeric statistics, fmt_fun= is the styling/formatting function. Default is NULL
header	Column header of new column. Default is "**Statistic**"
footnote	Footnote associated with new column. Default is no footnote (i.e. NULL)
new_col_name	name of new column to be created in .$table_body. Default is "add_stat_1", unless that column exists then it is "add_stat_2", etc.

Details

The custom functions passed in fns= are required to follow a specified format. Each of these function will execute on a single variable from `tbl_summary()`.

1. Each function must return a single scalar or character value of length one.
2. Each function may take the following arguments: `foo(data,variable,by,tbl)`

- data= is the input data frame passed to `tbl_summary()`
- variable= is a string indicating the variable to perform the calculation on
- by= is a string indicating the by variable from tbl_summary=, if present
- tbl= the original `tbl_summary()` object is also available to utilize

The user-defined does not need to utilize each of these inputs. It's encouraged the user-defined function accept `...` as each of the arguments *will* be passed to the function, even if not all inputs are utilized by the user's function, e.g. `foo(data,variable,by,...)`

Example Output

Examples

```
# Example 1 ----------------------------------
# this example replicates `add_p()`

# fn returns t-test pvalue
my_ttest <- function(data, variable, by, ...) {
  t.test(data[[variable]] ~ as.factor(data[[by]]))$p.value
}

add_stat_ex1 <-
  trial %>%
  select(trt, age, marker) %>%
  tbl_summary(by = trt, missing = "no") %>%
  add_p(test = everything() ~ t.test) %>%
  # replicating result of `add_p()` with `add_stat()`
  add_stat(
    fns = everything() ~ my_ttest, # all variables compared with with t-test
    fmt_fun = style_pvalue,        # format result with style_pvalue()
    header = "**My p-value**"      # new column header
  )

# Example 2 ----------------------------------
# fn returns t-test test statistic and pvalue
my_ttest2 <- function(data, variable, by, ...) {
  tt <- t.test(data[[variable]] ~ as.factor(data[[by]]))

  # returning test statistic and pvalue
  stringr::str_glue(
    "t={style_sigfig(tt$statistic)}, {style_pvalue(tt$p.value, prepend_p = TRUE)}"
  )
}

add_stat_ex2 <-
  trial %>%
```

```
  select(trt, age, marker) %>%
  tbl_summary(by = trt, missing = "no") %>%
  add_stat(
    fns = everything() ~ my_ttest2,    # all variables will be compared by t-test
    fmt_fun = NULL, # fn returns and chr, so no formatting function needed
    header = "**Treatment Comparison**",       # column header
    footnote = "T-test statistic and p-value"  # footnote
  )
# Example 1 ---------------------------------
```

`add_stat_label` *Add statistic labels*

Description

Adds labels describing the summary statistics presented for each variable in the tbl_summary / tbl_svysummary table.

Usage

```
add_stat_label(x, location = NULL, label = NULL)
```

Arguments

`x`	Object with class `tbl_summary` from the tbl_summary function or with class `tbl_svysummary` from the tbl_svysummary function
`location`	location where statistic label will be included. `"row"` (the default) to add the statistic label to the variable label row, and `"column"` adds a column with the statistic label.
`label`	a list of formulas or a single formula updating the statistic label, e.g. `label = all_categorical() ~ "No. (%)"`

Value

A `tbl_summary` or `tbl_svysummary` object

Example Output

Author(s)

Daniel D. Sjoberg

See Also

Other tbl_summary tools: `add_n.tbl_summary()`, `add_overall()`, `add_p.tbl_summary()`, `add_q()`, `bold_italicize_labels_levels`, `inline_text.tbl_summary()`, `inline_text.tbl_survfit()`, `modify`, `tbl_merge()`, `tbl_stack()`, `tbl_summary()`

Other tbl_svysummary tools: `add_n.tbl_summary()`, `add_overall()`, `add_p.tbl_svysummary()`, `add_q()`, `modify`, `tbl_merge()`, `tbl_stack()`, `tbl_svysummary()`

Examples

```
tbl <- trial %>%
  dplyr::select(trt, age, grade, response) %>%
  tbl_summary(by = trt)

# Example 1 ----------------------------------
# Add statistic presented to the variable label row
add_stat_label_ex1 <-
  tbl %>%
  add_stat_label(
    # update default statistic label for continuous variables
    label = all_continuous() ~ "med. (iqr)"
  )

# Example 2 ----------------------------------
add_stat_label_ex2 <-
  tbl %>%
  add_stat_label(
    # add a new column with statistic labels
    location = "column"
  )
```

---

as_flex_table *Convert gtsummary object to a flextable object*

Description

Function converts a gtsummary object to a flextable object. A user can use this function if they wish to add customized formatting available via the flextable functions. The flextable output is particularly useful when combined with R markdown with Word output, since the gt package does not support Word.

Usage

```
as_flex_table(
  x,
  include = everything(),
  return_calls = FALSE,
  strip_md_bold = TRUE
)
```

Arguments

x	Object created by a function from the gtsummary package (e.g. tbl_summary or tbl_regression)
include	Commands to include in output. Input may be a vector of quoted or unquoted names. tidyselect and gtsummary select helper functions are also accepted. Default is everything().
return_calls	Logical. Default is FALSE. If TRUE, the calls are returned as a list of expressions.
strip_md_bold	When TRUE, all double asterisk (markdown language for bold weight) in column labels and spanning headers are removed. Default is TRUE

Value

A flextable object

Details

The `as_flex_table()` functions converts the gtsummary object to a flextable, and prints it with the following styling functions.

1. `flextable::flextable()`
2. `flextable::set_header_labels()` to set column labels
3. `flextable::add_header_row()`, if applicable, to set spanning column header
4. `flextable::align()` to set column alignment
5. `flextable::padding()` to indent variable levels
6. `flextable::fontsize()` to set font size
7. `flextable::autofit()` to estimate the column widths
8. `flextable::footnote()` to add table footnotes and source notes
9. `flextable::bold()` to bold cells in data frame
10. `flextable::italic()` to italicize cells in data frame
11. `flextable::border()` to set all border widths to 1
12. `flextable::padding()` to set consistent header padding
13. `flextable::valign()` to ensure label column is top-left justified

Any one of these commands may be omitted using the include= argument.

Pro tip: Use the `flextable::width()` function for exacting control over column width after calling `as_flex_table()`.

Example Output

Author(s)

Daniel D. Sjoberg

See Also

Other gtsummary output types: `as_gt()`, `as_hux_table()`, `as_kable_extra()`, `as_kable()`, `as_tibble.gtsummary()`

Examples

```
as_flex_table_ex1 <-
  trial %>%
  select(trt, age, grade) %>%
  tbl_summary(by = trt) %>%
  add_p() %>%
  as_flex_table()
```

as_gt *Convert gtsummary object to a gt object*

Description

Function converts a gtsummary object to a gt_tbl object. Function is used in the background when the results are printed or knit. A user can use this function if they wish to add customized formatting available via the gt package.

Review the tbl_summary vignette or tbl_regression vignette for detailed examples in the 'Advanced Customization' section.

Usage

```
as_gt(
  x,
  include = everything(),
  return_calls = FALSE,
  exclude = NULL,
  omit = NULL
)
```

Arguments

x	Object created by a function from the gtsummary package (e.g. tbl_summary or tbl_regression)
include	Commands to include in output. Input may be a vector of quoted or unquoted names. tidyselect and gtsummary select helper functions are also accepted. Default is everything().
return_calls	Logical. Default is FALSE. If TRUE, the calls are returned as a list of expressions.
exclude	DEPRECATED.
omit	DEPRECATED.

Value

A gt_tbl object

Example Output

Author(s)

Daniel D. Sjoberg

See Also

Other gtsummary output types: as_flex_table(), as_hux_table(), as_kable_extra(), as_kable(), as_tibble.gtsummary()

Examples

```
as_gt_ex <-
  trial[c("trt", "age", "response", "grade")] %>%
  tbl_summary(by = trt) %>%
  as_gt()
```

`as_hux_table` *Convert gtsummary object to a huxtable object*

Description

Experimental Function converts a gtsummary object to a huxtable object. A user can use this function if they wish to add customized formatting available via the huxtable functions. The huxtable package supports output to PDF via LaTeX, as well as HTML and Word.

Usage

```
as_hux_table(
  x,
  include = everything(),
  return_calls = FALSE,
  strip_md_bold = TRUE
)
```

Arguments

`x`	Object created by a function from the gtsummary package (e.g. tbl_summary or tbl_regression)
`include`	Commands to include in output. Input may be a vector of quoted or unquoted names. tidyselect and gtsummary select helper functions are also accepted. Default is `everything()`.
`return_calls`	Logical. Default is `FALSE`. If `TRUE`, the calls are returned as a list of expressions.
`strip_md_bold`	When TRUE, all double asterisk (markdown language for bold weight) in column labels and spanning headers are removed. Default is TRUE

Value

A huxtable object

Details

The `as_hux_table()` takes the data frame that will be printed, converts it to a huxtable and formats the table with the following huxtable functions:

1. `huxtable::huxtable()`
2. `huxtable::insert_row()` to insert header rows
3. `huxtable::align()` to set column alignment
4. `huxtable::set_left_padding()` to indent variable levels
5. `huxtable::add_footnote()` to add table footnotes and source notes

6. huxtable::set_bold() to bold cells
7. huxtable::set_italic() to italicize cells
8. huxtable::set_na_string() to use an em-dash for missing numbers

Any one of these commands may be omitted using the include= argument.

Author(s)

David Hugh-Jones

See Also

Other gtsummary output types: as_flex_table(), as_gt(), as_kable_extra(), as_kable(), as_tibble.gtsummary()

Examples

```
trial %>%
  dplyr::select(trt, age, grade) %>%
  tbl_summary(by = trt) %>%
  add_p() %>%
  as_hux_table()
```

---

as_kable *Convert gtsummary object to a kable object*

Description

Function converts a gtsummary object to a knitr_kable object. This function is used in the background when the results are printed or knit. A user can use this function if they wish to add customized formatting available via knitr::kable.

Output from knitr::kable is less full featured compared to summary tables produced with gt. For example, kable summary tables do not include indentation, footnotes, or spanning header rows.

Usage

```
as_kable(x, include = everything(), return_calls = FALSE, exclude = NULL, ...)
```

Arguments

x	Object created by a function from the gtsummary package (e.g. tbl_summary or tbl_regression)
include	Commands to include in output. Input may be a vector of quoted or unquoted names. tidyselect and gtsummary select helper functions are also accepted. Default is everything().
return_calls	Logical. Default is FALSE. If TRUE, the calls are returned as a list of expressions.
exclude	DEPRECATED
...	Additional arguments passed to knitr::kable

Details

Tip: To better distinguish variable labels and level labels when indenting is not supported, try `bold_labels()` or `italicize_levels()`.

Value

A `knitr_kable` object

Author(s)

Daniel D. Sjoberg

See Also

Other gtsummary output types: `as_flex_table()`, `as_gt()`, `as_hux_table()`, `as_kable_extra()`, `as_tibble.gtsummary()`

Examples

```
trial %>%
  tbl_summary(by = trt) %>%
  bold_labels() %>%
  as_kable()
```

---

as_kable_extra *Convert gtsummary object to a kableExtra object*

Description

Experimental Function converts a gtsummary object to a knitr_kable + kableExtra object. A user can use this function if they wish to add customized formatting available via knitr::kable and kableExtra. Note that gtsummary uses the standard markdown ** to bold headers, and they may need to be changed manually with kableExtra output.

Usage

```
as_kable_extra(
  x,
  include = everything(),
  return_calls = FALSE,
  strip_md_bold = TRUE,
  ...
)
```

Arguments

x	Object created by a function from the gtsummary package (e.g. tbl_summary or tbl_regression)
include	Commands to include in output. Input may be a vector of quoted or unquoted names. tidyselect and gtsummary select helper functions are also accepted. Default is `everything()`.

return_calls	Logical. Default is FALSE. If TRUE, the calls are returned as a list of expressions.
strip_md_bold	When TRUE, all double asterisk (markdown language for bold weight) in column labels and spanning headers are removed. Default is TRUE
...	Additional arguments passed to knitr::kable

Value

A kableExtra object

Author(s)

Daniel D. Sjoberg

See Also

Other gtsummary output types: as_flex_table(), as_gt(), as_hux_table(), as_kable(), as_tibble.gtsummary()

Examples

```
tbl <-
  trial %>%
  tbl_summary(by = trt) %>%
  as_kable_extra()
```

---

as_tibble.gtsummary *Convert gtsummary object to a tibble*

Description

Function converts a gtsummary object to a tibble.

Usage

```
## S3 method for class 'gtsummary'
as_tibble(
  x,
  include = everything(),
  col_labels = TRUE,
  return_calls = FALSE,
  exclude = NULL,
  ...
)
```

Arguments

x	Object created by a function from the gtsummary package (e.g. tbl_summary or tbl_regression)
include	Commands to include in output. Input may be a vector of quoted or unquoted names. tidyselect and gtsummary select helper functions are also accepted. Default is everything().
col_labels	Logical argument adding column labels to output tibble. Default is TRUE.

return_calls	Logical. Default is FALSE. If TRUE, the calls are returned as a list of expressions.
exclude	DEPRECATED
...	Not used

Value

a tibble

Author(s)

Daniel D. Sjoberg

See Also

Other gtsummary output types: as_flex_table(), as_gt(), as_hux_table(), as_kable_extra(), as_kable()

Examples

```
tbl <-
  trial %>%
  select(trt, age, grade, response) %>%
  tbl_summary(by = trt)

as_tibble(tbl)

# without column labels
as_tibble(tbl, col_labels = FALSE)
```

---

bold_italicize_labels_levels

Bold or Italicize labels or levels in gtsummary tables

Description

Bold or Italicize labels or levels in gtsummary tables

Usage

```
bold_labels(x)

bold_levels(x)

italicize_labels(x)

italicize_levels(x)
```

Arguments

x	Object created using gtsummary functions

Value

Functions return the same class of gtsummary object supplied

Functions

- `bold_labels`: Bold labels in gtsummary tables
- `bold_levels`: Bold levels in gtsummary tables
- `italicize_labels`: Italicize labels in gtsummary tables
- `italicize_levels`: Italicize levels in gtsummary tables

Example Output

Author(s)

Daniel D. Sjoberg

See Also

Other tbl_summary tools: `add_n.tbl_summary()`, `add_overall()`, `add_p.tbl_summary()`, `add_q()`, `add_stat_label()`, `inline_text.tbl_summary()`, `inline_text.tbl_survfit()`, `modify`, `tbl_merge()`, `tbl_stack()`, `tbl_summary()`

Other tbl_regression tools: `add_global_p.tbl_regression()`, `add_nevent.tbl_regression()`, `add_q()`, `combine_terms()`, `inline_text.tbl_regression()`, `modify`, `tbl_merge()`, `tbl_regression()`, `tbl_stack()`

Other tbl_uvregression tools: `add_global_p.tbl_uvregression()`, `add_nevent.tbl_uvregression()`, `add_q()`, `inline_text.tbl_uvregression()`, `modify`, `tbl_merge()`, `tbl_stack()`, `tbl_uvregression()`

Examples

```
tbl_bold_ital_ex <-
  trial[c("trt", "age", "grade")] %>%
  tbl_summary() %>%
  bold_labels() %>%
  bold_levels() %>%
  italicize_labels() %>%
  italicize_levels()
```

---

bold_p *Bold significant p-values or q-values*

Description

Bold values below a chosen threshold (e.g. <0.05) in a gtsummary tables.

Usage

```
bold_p(x, t = 0.05, q = FALSE)
```

Arguments

x	Object created using gtsummary functions
t	Threshold below which values will be bold. Default is 0.05.
q	Logical argument. When TRUE will bold the q-value column rather than the p-values. Default is FALSE.

Example Output

Author(s)

Daniel D. Sjoberg, Esther Drill

Examples

```
# Example 1 ----------------------------------
bold_p_ex1 <-
  trial[c("age", "grade", "response", "trt")] %>%
  tbl_summary(by = trt) %>%
  add_p() %>%
  bold_p(t = 0.65)

# Example 2 ----------------------------------
bold_p_ex2 <-
  glm(response ~ trt + grade, trial, family = binomial(link = "logit")) %>%
  tbl_regression(exponentiate = TRUE) %>%
  bold_p(t = 0.65)
```

---

combine_terms *Combine terms in a regression model*

Description

Experimental The function combines terms from a regression model, and replaces the terms with a single row in the output table. The p-value is calculated using `stats::anova()`.

Usage

```
combine_terms(x, formula_update, label = NULL, quiet = NULL, ...)
```

Arguments

x	a tbl_regression object
formula_update	formula update passed to the stats::update. This updated formula is used to construct a reduced model, and is subsequently passed to `stats::anova()` to calculate the p-value for the group of removed terms. See the stats::update help file for proper syntax. function's formula.= argument
label	Option string argument labeling the combined rows
quiet	Logical indicating whether to print messages in console. Default is FALSE
...	Additional arguments passed to stats::anova

Value

tbl_regression object

Example Output

Author(s)

Daniel D. Sjoberg

See Also

Other tbl_regression tools: add_global_p.tbl_regression(), add_nevent.tbl_regression(), add_q(), bold_italicize_labels_levels, inline_text.tbl_regression(), modify, tbl_merge(), tbl_regression(), tbl_stack()

Examples

```
# Example 1 ----------------------------------
# fit model with nonlinear terms for marker
nlmod1 <- lm(
  age ~ marker + I(marker^2) + grade,
  trial[c("age", "marker", "grade")] %>% na.omit() # keep complete cases only!
)

combine_terms_ex1 <-
  tbl_regression(nlmod1, label = grade ~ "Grade") %>%
  # collapse non-linear terms to a single row in output using anova
  combine_terms(
    formula_update = . ~ . - marker - I(marker^2),
    label = "Marker (non-linear terms)"
  )

# Example 2 ----------------------------------
# Example with Cubic Splines
library(Hmisc, warn.conflicts = FALSE, quietly = TRUE)
mod2 <- lm(
  age ~ rcspline.eval(marker, inclx = TRUE) + grade,
  trial[c("age", "marker", "grade")] %>% na.omit() # keep complete cases only!
)

combine_terms_ex2 <-
  tbl_regression(mod2, label = grade ~ "Grade") %>%
  combine_terms(
    formula_update = . ~ . -rcspline.eval(marker, inclx = TRUE),
    label = "Marker (non-linear terms)"
  )

# Example 3 ----------------------------------
# Logistic Regression Example, LRT p-value
combine_terms_ex3 <-
  glm(
    response ~ marker + I(marker^2) + grade,
    trial[c("response", "marker", "grade")] %>% na.omit(), # keep complete cases only!
    family = binomial
```

```
) %>%
tbl_regression(label = grade ~ "Grade", exponentiate = TRUE) %>%
# collapse non-linear terms to a single row in output using anova
combine_terms(
  formula_update = . ~ . - marker - I(marker^2),
  label = "Marker (non-linear terms)",
  test = "LRT"
)
```

`inline_text` *Report statistics from gtsummary tables inline*

Description

Report statistics from gtsummary tables inline

Usage

```
inline_text(x, ...)
```

Arguments

`x`	Object created from a gtsummary function
`...`	Additional arguments passed to other methods.

Value

A string reporting results from a gtsummary table

Author(s)

Daniel D. Sjoberg

See Also

inline_text.tbl_summary, inline_text.tbl_regression, inline_text.tbl_uvregression, inline_text.tbl_survfit

---

`inline_text.tbl_cross` *Report statistics from cross table inline*

Description

Experimental Extracts and returns statistics from a `tbl_cross` object for inline reporting in an R markdown document. Detailed examples in the inline_text vignette

Usage

```
## S3 method for class 'tbl_cross'
inline_text(x, col_level = NULL, row_level = NULL, pvalue_fun = NULL, ...)
```

Arguments

x	a tbl_cross object
col_level	Level of the column variable to display. Default is NULL Can also specify "p.value" for the p-value and "stat_0" for Total column.
row_level	Level of the row variable to display. Can also specify the 'Unknown' row. Default is NULL
pvalue_fun	Function to round and format p-values. Default is style_pvalue. The function must have a numeric vector input (the numeric, exact p-value), and return a string that is the rounded/formatted p-value (e.g. pvalue_fun = function(x) style_pvalue(x,digits = 2) or equivalently, purrr::partial(style_pvalue,digits = 2)).
...	Not used

Value

A string reporting results from a gtsummary table

See Also

Other tbl_cross tools: add_p.tbl_cross(), tbl_cross()

Examples

```
tbl_cross <-
  tbl_cross(trial, row = trt, col = response) %>%
  add_p()

inline_text(tbl_cross, row_level = "Drug A", col_level = "1")
inline_text(tbl_cross, row_level = "Total", col_level = "1")
inline_text(tbl_cross, col_level = "p.value")
```

---

inline_text.tbl_regression

Report statistics from regression summary tables inline

Description

Takes an object with class tbl_regression, and the location of the statistic to report and returns statistics for reporting inline in an R markdown document. Detailed examples in the inline_text vignette

Usage

```
## S3 method for class 'tbl_regression'
inline_text(
  x,
  variable,
  level = NULL,
 pattern = "{estimate} ({conf.level*100}% CI {conf.low}, {conf.high}; {p.value})",
  estimate_fun = NULL,
```

```
  pvalue_fun = NULL,
  ...
)
```

Arguments

x	Object created from tbl_regression
variable	Variable name of statistics to present
level	Level of the variable to display for categorical variables. Default is NULL, returning the top row in the table for the variable.
pattern	String indicating the statistics to return. Uses glue::glue formatting. Default is "{estimate} ({conf.level }% CI {conf.low},{conf.high}; {p.value})". All columns from x$table_body are available to print as well as the confidence level (conf.level). See below for details.
estimate_fun	function to style model coefficient estimates. Columns 'estimate', 'conf.low', and 'conf.high' are formatted. Default is x$inputs$estimate_fun
pvalue_fun	function to style p-values and/or q-values. Default is function(x) style_pvalue(x,prepend_p = TRUE)
...	Not used

Value

A string reporting results from a gtsummary table

pattern argument

The following items are available to print. Use print(x$table_body) to print the table the estimates are extracted from.

- {estimate} coefficient estimate formatted with 'estimate_fun'
- {conf.low} lower limit of confidence interval formatted with 'estimate_fun'
- {conf.high} upper limit of confidence interval formatted with 'estimate_fun'
- {ci} confidence interval formatted with x$estimate_fun
- {p.value} p-value formatted with 'pvalue_fun'
- {N} number of observations in model
- {label} variable/variable level label

Author(s)

Daniel D. Sjoberg

See Also

Other tbl_regression tools: add_global_p.tbl_regression(), add_nevent.tbl_regression(), add_q(), bold_italicize_labels_levels, combine_terms(), modify, tbl_merge(), tbl_regression(), tbl_stack()

Examples

```
inline_text_ex1 <-
  glm(response ~ age + grade, trial, family = binomial(link = "logit")) %>%
  tbl_regression(exponentiate = TRUE)

inline_text(inline_text_ex1, variable = age)
inline_text(inline_text_ex1, variable = grade, level = "III")
```

---

inline_text.tbl_summary

Report statistics from summary tables inline

Description

Extracts and returns statistics from a `tbl_summary` object for inline reporting in an R markdown document. Detailed examples in the inline_text vignette

Usage

```
## S3 method for class 'tbl_summary'
inline_text(
  x,
  variable,
  column = NULL,
  level = NULL,
  pattern = NULL,
  pvalue_fun = NULL,
  ...
)

## S3 method for class 'tbl_svysummary'
inline_text(
  x,
  variable,
  column = NULL,
  level = NULL,
  pattern = NULL,
  pvalue_fun = NULL,
  ...
)
```

Arguments

`x`	Object created from tbl_summary
`variable`	Variable name of statistic to present
`column`	Column name to return from `x$table_body`. Can also pass the level of a by variable.
`level`	Level of the variable to display for categorical variables. Can also specify the 'Unknown' row. Default is `NULL`

pattern	String indicating the statistics to return. Uses glue::glue formatting. Default is pattern shown in tbl_summary() output
pvalue_fun	Function to round and format p-values. Default is style_pvalue. The function must have a numeric vector input (the numeric, exact p-value), and return a string that is the rounded/formatted p-value (e.g. pvalue_fun = function(x) style_pvalue(x,digits = 2) or equivalently, purrr::partial(style_pvalue,digits = 2)).
...	Not used

Value

A string reporting results from a gtsummary table

Author(s)

Daniel D. Sjoberg

See Also

Other tbl_summary tools: add_n.tbl_summary(), add_overall(), add_p.tbl_summary(), add_q(), add_stat_label(), bold_italicize_labels_levels, inline_text.tbl_survfit(), modify, tbl_merge(), tbl_stack(), tbl_summary()

Examples

```
t1 <- trial[c("trt", "grade")] %>% tbl_summary(by = trt) %>% add_p()

inline_text(t1, variable = grade, level = "I", column = "Drug A", pattern = "{n}/{N} ({p})%")
inline_text(t1, variable = grade, column = "p.value")
```

---

inline_text.tbl_survfit

Report statistics from survfit tables inline

Description

Experimental Extracts and returns statistics from a tbl_survfit object for inline reporting in an R markdown document. Detailed examples in the inline_text vignette

Usage

```
## S3 method for class 'tbl_survfit'
inline_text(
  x,
  time = NULL,
  prob = NULL,
  variable = NULL,
  level = NULL,
  pattern = x$inputs$statistic,
  estimate_fun = x$inputs$estimate_fun,
  pvalue_fun = NULL,
  ...
)
```

Arguments

x	Object created from tbl_survfit
time	time for which to return survival probabilities.
prob	probability with values in (0,1)
variable	Variable name of statistic to present.
level	Level of the variable to display for categorical variables. Can also specify the 'Unknown' row. Default is NULL
pattern	String indicating the statistics to return.
estimate_fun	Function to round and format coefficient estimates. Default is style_sigfig when the coefficients are not transformed, and style_ratio when the coefficients have been exponentiated.
pvalue_fun	Function to round and format p-values. Default is style_pvalue. The function must have a numeric vector input (the numeric, exact p-value), and return a string that is the rounded/formatted p-value (e.g. pvalue_fun = function(x) style_pvalue(x,digits = 2) or equivalently, purrr::partial(style_pvalue,digits = 2)).
...	tbl_survfit used

Value

A string reporting results from a gtsummary table

Author(s)

Daniel D. Sjoberg

See Also

Other tbl_summary tools: add_n.tbl_summary(), add_overall(), add_p.tbl_summary(), add_q(), add_stat_label(), bold_italicize_labels_levels, inline_text.tbl_summary(), modify, tbl_merge(), tbl_stack(), tbl_summary()

Examples

```
library(survival)
# fit survfit
fit1 <- survfit(Surv(ttdeath, death) ~ trt, trial)
fit2 <- survfit(Surv(ttdeath, death) ~ 1, trial)

# sumarize survfit objects
tbl1 <- tbl_survfit(
  fit1,
  times = c(12, 24),
  label = "Treatment",
  label_header = "**{time} Month**"
)

tbl2 <- tbl_survfit(
  fit2,
  probs = 0.5,
  label_header = "**Median Survival**"
)
```

```
# report results inline
inline_text(tbl1, time = 24, level = "Drug B")
inline_text(tbl2, prob = 0.5)
```

`inline_text.tbl_uvregression` *Report statistics from regression summary tables inline*

Description

Extracts and returns statistics from a table created by the `tbl_uvregression` function for inline reporting in an R markdown document. Detailed examples in the inline_text vignette

Usage

```
## S3 method for class 'tbl_uvregression'
inline_text(
  x,
  variable,
  level = NULL,
 pattern = "{estimate} ({conf.level*100}% CI {conf.low}, {conf.high}; {p.value})",
  estimate_fun = NULL,
  pvalue_fun = NULL,
  ...
)
```

Arguments

`x`	Object created from tbl_uvregression
`variable`	Variable name of statistics to present
`level`	Level of the variable to display for categorical variables. Default is `NULL`, returning the top row in the table for the variable.
`pattern`	String indicating the statistics to return. Uses glue::glue formatting. Default is `"{estimate} ({conf.level }% CI {conf.low},{conf.high}; {p.value})"`. All columns from `x$table_body` are available to print as well as the confidence level (conf.level). See below for details.
`estimate_fun`	function to style model coefficient estimates. Columns 'estimate', 'conf.low', and 'conf.high' are formatted. Default is `x$inputs$estimate_fun`
`pvalue_fun`	function to style p-values and/or q-values. Default is `function(x) style_pvalue(x,prepend_p = TRUE)`
`...`	Not used

Value

A string reporting results from a gtsummary table

pattern argument

The following items are available to print. Use `print(x$table_body)` to print the table the estimates are extracted from.

- `{estimate}` coefficient estimate formatted with 'estimate_fun'
- `{conf.low}` lower limit of confidence interval formatted with 'estimate_fun'
- `{conf.high}` upper limit of confidence interval formatted with 'estimate_fun'
- `{ci}` confidence interval formatted with x$estimate_fun
- `{p.value}` p-value formatted with 'pvalue_fun'
- `{N}` number of observations in model
- `{label}` variable/variable level label

Author(s)

Daniel D. Sjoberg

See Also

Other tbl_uvregression tools: `add_global_p.tbl_uvregression()`, `add_nevent.tbl_uvregression()`, `add_q()`, `bold_italicize_labels_levels`, `modify`, `tbl_merge()`, `tbl_stack()`, `tbl_uvregression()`

Examples

```
inline_text_ex1 <-
  trial[c("response", "age", "grade")] %>%
  tbl_uvregression(
    method = glm,
    method.args = list(family = binomial),
    y = response,
    exponentiate = TRUE
  )

inline_text(inline_text_ex1, variable = age)
inline_text(inline_text_ex1, variable = grade, level = "III")
```

---

modify *Modify column headers, footnotes, and spanning headers*

Description

These functions assist with updating or adding column headers (`modify_header()`), footnotes (`modify_footnote()`), and spanning headers (`modify_spanning_header()`). Use `show_header_names()` to learn the column names.

Usage

```
modify_header(
  x,
  update = NULL,
  stat_by = NULL,
  text_interpret = c("md", "html"),
  ...
)

modify_footnote(x, update, abbreviation = FALSE)

modify_spanning_header(x, update)

show_header_names(x = NULL, quiet = NULL)
```

Arguments

x	a gtsummary object
update	list of formulas or a single formula specifying the updated column header, footnote, or spanning header. The LHS specifies the column(s) to be updated, and the RHS is the updated text. Use the `show_header_names()` to see the column names that can be modified.
stat_by	Used with `tbl_summary(by=)` objects with a by= argument. String specifying text to include above the summary statistics. The following fields are available for use in the headers: • {n} number of observations in each group, • {N} total number of observations, • {p} percentage in each group, • {level} the 'by' variable level, Syntax follows `glue::glue()`, e.g. `stat_by = "**{level}**,N = {n} ({style_percent(p)}%)"`.
text_interpret	String indicates whether text will be interpreted with `gt::md()` or `gt::html()`. Must be `"md"` (default) or `"html"`.
...	Specify a column and updated column label, e.g. `modify_header(p.value = "Model P-values")`. This is provided as an alternative to the update= argument. They accomplish the same goal of updating column headers.
abbreviation	Logical indicating if an abbreviation is being updated.
quiet	Logical indicating whether to print messages in console. Default is `FALSE`

Value

Updated gtsummary object

Example Output

Author(s)

Daniel D. Sjoberg

See Also

Other tbl_summary tools: add_n.tbl_summary(), add_overall(), add_p.tbl_summary(), add_q(), add_stat_label(), bold_italicize_labels_levels, inline_text.tbl_summary(), inline_text.tbl_survfit(), tbl_merge(), tbl_stack(), tbl_summary()

Other tbl_svysummary tools: add_n.tbl_summary(), add_overall(), add_p.tbl_svysummary(), add_q(), add_stat_label(), tbl_merge(), tbl_stack(), tbl_svysummary()

Other tbl_regression tools: add_global_p.tbl_regression(), add_nevent.tbl_regression(), add_q(), bold_italicize_labels_levels, combine_terms(), inline_text.tbl_regression(), tbl_merge(), tbl_regression(), tbl_stack()

Other tbl_uvregression tools: add_global_p.tbl_uvregression(), add_nevent.tbl_uvregression(), add_q(), bold_italicize_labels_levels, inline_text.tbl_uvregression(), tbl_merge(), tbl_stack(), tbl_uvregression()

Other tbl_survfit tools: add_n.tbl_survfit(), add_nevent.tbl_survfit(), add_p.tbl_survfit(), tbl_merge(), tbl_stack(), tbl_survfit()

Examples

```
# create summary table
tbl <- trial[c("age", "grade", "trt")] %>%
  tbl_summary(by = trt, missing = "no") %>%
  add_p()

# print the column names that can be modified
show_header_names(tbl)

# Example 1 ----------------------------------
# updating column headers and footnote
modify_ex1 <- tbl %>%
  modify_header(
    update = list(label ~ "**Variable**",
                  p.value ~ "**P**")
  ) %>%
  modify_footnote(
    update = starts_with("stat_") ~ "median (IQR) for Age; n (%) for Grade"
  )

# Example 2 ----------------------------------
# using `stat_by=` argument to update headers, remove all footnotes, add spanning header
modify_ex2 <- tbl %>%
  modify_header(stat_by = "**{level}**, N = {n} ({style_percent(p)}%)") %>%
 # use `modify_footnote(everything() ~ NA, abbreviation = TRUE)` to delete abbrev. footnotes
  modify_footnote(update = everything() ~ NA) %>%
  modify_spanning_header(starts_with("stat_") ~ "**Treatment Received**")

# Example 3 ----------------------------------
# updating an abbreviation in table footnote
modify_ex3 <-
  glm(response ~ age + grade, trial, family = binomial) %>%
  tbl_regression(exponentiate = TRUE) %>%
  modify_footnote(ci ~ "CI = Credible Interval", abbreviation = TRUE)
```

print_gtsummary *print and knit_print methods for gtsummary objects*

Description

print and knit_print methods for gtsummary objects

Usage

```
## S3 method for class 'gtsummary'
print(x, print_engine = NULL, ...)

## S3 method for class 'gtsummary'
knit_print(x, ...)
```

Arguments

x	An object created using gtsummary functions
print_engine	String indicating the print method. Must be one of "gt", "kable", "kable_extra", "flextable", "tibble"
...	Not used

Author(s)

Daniel D. Sjoberg

See Also

tbl_summary tbl_regression tbl_uvregression tbl_merge tbl_stack

select_helpers *Select helper functions*

Description

Set of functions to supplement the tidyselect set of functions for selecting columns of data frames. all_continuous(), all_categorical(), and all_dichotomous() may only be used with tbl_summary(), where each variable has been classified into one of these three groups. All other helpers are available throughout the package.

Usage

```
all_continuous()

all_categorical(dichotomous = TRUE)

all_dichotomous()

all_numeric()
```

```
all_character()

all_integer()

all_double()

all_logical()

all_factor()
```

Arguments

dichotomous	Logical indicating whether to include dichotomous variables. Default is TRUE

Value

A character vector of column names selected

Examples

```
select_ex1 <-
  trial %>%
  select(age, response, grade) %>%
  tbl_summary(
    statistic = all_continuous() ~ "{mean} ({sd})",
    type = all_dichotomous() ~ "categorical"
  )
```

---

set_gtsummary_theme *Set a gtsummary theme*

Description

Experimental Use this function to set preferences for the display of gtsummary tables. The default formatting and styling throughout the gtsummary package are taken from the published reporting guidelines of the top four urology journals: European Urology, The Journal of Urology, Urology and the British Journal of Urology International. Use this function to change the default reporting style to match another journal, or your own personal style.

Usage

```
set_gtsummary_theme(x)

reset_gtsummary_theme()
```

Arguments

x	A gtsummary theme function, e.g. theme_gtsummary_journal(), or a named list defining a gtsummary theme. See details below.

Example Output

See Also

Themes vignette

Available gtsummary themes

Examples

```
# Setting JAMA theme for gtsummary
set_gtsummary_theme(theme_gtsummary_journal("jama"))
# Themes can be combined by including more than one
set_gtsummary_theme(theme_gtsummary_compact())

set_gtsummary_theme_ex1 <-
  trial %>%
  dplyr::select(age, grade, trt) %>%
  tbl_summary(by = trt) %>%
  add_stat_label() %>%
  as_gt()

# reset gtsummary theme
reset_gtsummary_theme()
```

---

sort_p *Sort variables in table by ascending p-values*

Description

Sort tables created by gtsummary by p-values

Usage

```
sort_p(x, q = FALSE)
```

Arguments

x	An object created using gtsummary functions
q	Logical argument. When TRUE will sort by the q-value column

Example Output

Author(s)

Karissa Whiting

Examples

```
# Example 1 ----------------------------------
sort_p_ex1 <-
  trial[c("age", "grade", "response", "trt")] %>%
  tbl_summary(by = trt) %>%
  add_p() %>%
  sort_p()

# Example 2 ----------------------------------
sort_p_ex2 <-
  glm(response ~ trt + grade, trial, family = binomial(link = "logit")) %>%
  tbl_regression(exponentiate = TRUE) %>%
  sort_p()
```

`style_number` *Style numbers*

Description

Style numbers

Usage

```
style_number(
  x,
  digits = 0,
  big.mark = NULL,
  decimal.mark = NULL,
  scale = 1,
  ...
)
```

Arguments

`x`	Numeric vector
`digits`	Integer or vector of integers specifying the number of digits to round x=. When vector is passed, each integer is mapped 1:1 to the numeric values in `x`
`big.mark`	Character used between every 3 digits to separate hundreds/thousands/millions/etc. Default is `","`, except when `decimal.mark = ","` when the default is a space.
`decimal.mark`	The character to be used to indicate the numeric decimal point. Default is `"."` or `getOption("OutDec")`
`scale`	A scaling factor: x will be multiplied by scale before formatting.
`...`	Other arguments passed on to `base::format()`

Value

formatted character vector

See Also

Other style tools: `style_percent()`, `style_pvalue()`, `style_ratio()`, `style_sigfig()`

Examples

```
c(0.111, 12.3) %>% style_number(digits = 1)
c(0.111, 12.3) %>% style_number(digits = c(1, 0))
```

`style_percent` *Style percentages*

Description

Style percentages

Usage

```
style_percent(x, symbol = FALSE, big.mark = NULL, decimal.mark = NULL, ...)
```

Arguments

`x`	numeric vector of percentages
`symbol`	Logical indicator to include percent symbol in output. Default is `FALSE`.
`big.mark`	Character used between every 3 digits to separate hundreds/thousands/millions/etc. Default is `","`, except when `decimal.mark = ","` when the default is a space.
`decimal.mark`	The character to be used to indicate the numeric decimal point. Default is `"."` or `getOption("OutDec")`
`...`	Other arguments passed on to `base::format()`

Value

A character vector of styled percentages

Author(s)

Daniel D. Sjoberg

See Also

See Table Gallery vignette for example

Other style tools: `style_number()`, `style_pvalue()`, `style_ratio()`, `style_sigfig()`

Examples

```
percent_vals <- c(-1, 0, 0.0001, 0.005, 0.01, 0.10, 0.45356, 0.99, 1.45)
style_percent(percent_vals)
style_percent(percent_vals, symbol = TRUE)
```

`style_pvalue` *Style p-values*

Description

Style p-values

Usage

```
style_pvalue(
  x,
  digits = 1,
  prepend_p = FALSE,
  big.mark = NULL,
  decimal.mark = NULL,
  ...
)
```

Arguments

`x`	Numeric vector of p-values.
`digits`	Number of digits large p-values are rounded. Must be 1, 2, or 3. Default is 1.
`prepend_p`	Logical. Should 'p=' be prepended to formatted p-value. Default is `FALSE`
`big.mark`	Character used between every 3 digits to separate hundreds/thousands/millions/etc. Default is `","`, except when `decimal.mark = ","` when the default is a space.
`decimal.mark`	The character to be used to indicate the numeric decimal point. Default is `"."` or `getOption("OutDec")`
`...`	Other arguments passed on to `base::format()`

Value

A character vector of styled p-values

Author(s)

Daniel D. Sjoberg

See Also

See tbl_summary vignette for examples

Other style tools: `style_number()`, `style_percent()`, `style_ratio()`, `style_sigfig()`

Examples

```
pvals <- c(
  1.5, 1, 0.999, 0.5, 0.25, 0.2, 0.197, 0.12, 0.10, 0.0999, 0.06,
  0.03, 0.002, 0.001, 0.00099, 0.0002, 0.00002, -1
)
style_pvalue(pvals)
style_pvalue(pvals, digits = 2, prepend_p = TRUE)
```

style_ratio *Style significant figure-like rounding for ratios*

Description

When reporting ratios, such as relative risk or an odds ratio, we'll often want the rounding to be similar on each side of the number 1. For example, if we report an odds ratio of 0.95 with a confidence interval of 0.70 to 1.24, we would want to round to two decimal places for all values. In other words, 2 significant figures for numbers less than 1 and 3 significant figures 1 and larger. `style_ratio()` performs significant figure-like rounding in this manner.

Usage

```
style_ratio(x, digits = 2, big.mark = NULL, decimal.mark = NULL, ...)
```

Arguments

`x`	Numeric vector
`digits`	Integer specifying the number of significant digits to display for numbers below 1. Numbers larger than 1 will be be `digits + 1`. Default is `digits = 2`.
`big.mark`	Character used between every 3 digits to separate hundreds/thousands/millions/etc. Default is `","`, except when `decimal.mark = ","` when the default is a space.
`decimal.mark`	The character to be used to indicate the numeric decimal point. Default is `"."` or `getOption("OutDec")`
`...`	Other arguments passed on to `base::format()`

Value

A character vector of styled ratios

Author(s)

Daniel D. Sjoberg

See Also

Other style tools: `style_number()`, `style_percent()`, `style_pvalue()`, `style_sigfig()`

Examples

```
c(
  0.123, 0.9, 1.1234, 12.345, 101.234, -0.123,
  -0.9, -1.1234, -12.345, -101.234
) %>%
  style_ratio()
```

`style_sigfig` *Style significant figure-like rounding*

Description

Converts a numeric argument into a string that has been rounded to a significant figure-like number. Scientific notation output is avoided, however, and additional significant figures may be displayed for large numbers. For example, if the number of significant digits requested is 2, 123 will be displayed (rather than 120 or 1.2x10^2).

Usage

```
style_sigfig(x, digits = 2, big.mark = NULL, decimal.mark = NULL, ...)
```

Arguments

`x`	Numeric vector
`digits`	Integer specifying the minimum number of significant digits to display
`big.mark`	Character used between every 3 digits to separate hundreds/thousands/millions/etc. Default is `","`, except when `decimal.mark = ","` when the default is a space.
`decimal.mark`	The character to be used to indicate the numeric decimal point. Default is `"."` or `getOption("OutDec")`
`...`	Other arguments passed on to `base::format()`

Details

If 2 sig figs are input, the number is rounded to 2 decimal places when `abs(x) < 1`, 1 decimal place when `abs(x) >= 1 & abs(x) < 10`, and to the nearest integer when `abs(x) >= 10`.

Value

A character vector of styled numbers

Author(s)

Daniel D. Sjoberg

See Also

Other style tools: `style_number()`, `style_percent()`, `style_pvalue()`, `style_ratio()`

Examples

```
c(0.123, 0.9, 1.1234, 12.345, -0.123, -0.9, -1.1234, -132.345, NA, -0.001) %>%
  style_sigfig()
```

tbl_cross *Create a cross table of summary statistics*

Description

Experimental The function creates a cross table of two categorical variables.

Usage

```
tbl_cross(
  data,
  row = NULL,
  col = NULL,
  label = NULL,
  statistic = NULL,
  percent = c("none", "column", "row", "cell"),
  margin = c("column", "row"),
  missing = c("ifany", "always", "no"),
  missing_text = "Unknown",
  margin_text = "Total"
)
```

Arguments

data	A data frame
row	A column name in data to be used for columns of cross table.
col	A column name in data to be used for rows of cross table.
label	List of formulas specifying variables labels, e.g. list(age ~ "Age",stage ~ "Path T Stage"). If a variable's label is not specified here, the label attribute (attr(data$age,"label")) is used. If attribute label is NULL, the variable name will be used.
statistic	A string with the statistic name in curly brackets to be replaced with the numeric statistic (see glue::glue). The default is {n}. If percent argument is "column", "row", or "cell", default is "{n} ({p}%)".
percent	Indicates the type of percentage to return. Must be one of "none", "column", "row", or "cell". Default is "cell" when {N} or {p} is used in statistic.
margin	Indicates which margins to add to the table. Default is c("row","column")
missing	Indicates whether to include counts of NA values in the table. Allowed values are "no" (never display NA values), "ifany" (only display if any NA values), and "always" (includes NA count row for all variables). Default is "ifany".
missing_text	String to display for count of missing observations. Default is "Unknown".
margin_text	Text to display for margin totals. Default is "Total"

Value

A tbl_cross object

Example Output

Author(s)

Karissa Whiting, Daniel D. Sjoberg

See Also

Other tbl_cross tools: `add_p.tbl_cross()`, `inline_text.tbl_cross()`

Examples

```
# Example 1 ----------------------------------
tbl_cross_ex1 <-
  trial %>%
  tbl_cross(row = trt, col = response)

# Example 2 ----------------------------------
tbl_cross_ex2 <-
  trial %>%
  tbl_cross(row = stage, col = trt, percent = "cell") %>%
  add_p()
```

---

tbl_merge *Merge two or more gtsummary objects*

Description

Merges two or more `tbl_regression`, `tbl_uvregression`, `tbl_stack`, `tbl_summary`, or `tbl_svysummary` objects and adds appropriate spanning headers.

Usage

```
tbl_merge(tbls, tab_spanner = NULL)
```

Arguments

`tbls`	List of gtsummary objects to merge
`tab_spanner`	Character vector specifying the spanning headers. Must be the same length as `tbls`. The strings are interpreted with `gt::md`. Must be same length as `tbls` argument

Value

A `tbl_merge` object

Example Output

Author(s)

Daniel D. Sjoberg

See Also

tbl_stack

Other tbl_regression tools: add_global_p.tbl_regression(), add_nevent.tbl_regression(), add_q(), bold_italicize_labels_levels, combine_terms(), inline_text.tbl_regression(), modify, tbl_regression(), tbl_stack()

Other tbl_uvregression tools: add_global_p.tbl_uvregression(), add_nevent.tbl_uvregression(), add_q(), bold_italicize_labels_levels, inline_text.tbl_uvregression(), modify, tbl_stack(), tbl_uvregression()

Other tbl_summary tools: add_n.tbl_summary(), add_overall(), add_p.tbl_summary(), add_q(), add_stat_label(), bold_italicize_labels_levels, inline_text.tbl_summary(), inline_text.tbl_survfit(modify, tbl_stack(), tbl_summary()

Other tbl_survfit tools: add_n.tbl_survfit(), add_nevent.tbl_survfit(), add_p.tbl_survfit(), modify, tbl_stack(), tbl_survfit()

Other tbl_svysummary tools: add_n.tbl_summary(), add_overall(), add_p.tbl_svysummary(), add_q(), add_stat_label(), modify, tbl_stack(), tbl_svysummary()

Examples

```
# Example 1 ----------------------------------
# Side-by-side Regression Models
library(survival)
t1 <-
  glm(response ~ trt + grade + age, trial, family = binomial) %>%
  tbl_regression(exponentiate = TRUE)
t2 <-
  coxph(Surv(ttdeath, death) ~ trt + grade + age, trial) %>%
  tbl_regression(exponentiate = TRUE)
tbl_merge_ex1 <-
  tbl_merge(
    tbls = list(t1, t2),
    tab_spanner = c("**Tumor Response**", "**Time to Death**")
  )

# Example 2 ----------------------------------
# Descriptive statistics alongside univariate regression, with no spanning header
t3 <-
  trial[c("age", "grade", "response")] %>%
  tbl_summary(missing = "no") %>%
  add_n %>%
  modify_header(stat_0 ~ "**Summary Statistics**")
t4 <-
  tbl_uvregression(
    trial[c("ttdeath", "death", "age", "grade", "response")],
    method = coxph,
    y = Surv(ttdeath, death),
    exponentiate = TRUE,
    hide_n = TRUE
  )
```

```
tbl_merge_ex2 <-
  tbl_merge(tbls = list(t3, t4)) %>%
  modify_spanning_header(everything() ~ NA_character_)
```

`tbl_regression` *Display regression model results in table*

Description

This function takes a regression model object and returns a formatted table that is publication-ready. The function is highly customizable allowing the user to obtain a bespoke summary table of the regression model results. Review the tbl_regression vignette for detailed examples.

Usage

```
tbl_regression(
  x,
  label = NULL,
  exponentiate = FALSE,
  include = everything(),
  show_single_row = NULL,
  conf.level = NULL,
  intercept = FALSE,
  estimate_fun = NULL,
  pvalue_fun = NULL,
  tidy_fun = NULL,
  show_yesno = NULL,
  exclude = NULL
)
```

Arguments

`x`	Regression model object
`label`	List of formulas specifying variables labels, e.g. `list(age ~ "Age", stage ~ "Path T Stage")`
`exponentiate`	Logical indicating whether to exponentiate the coefficient estimates. Default is `FALSE`.
`include`	Variables to include in output. Input may be a vector of quoted variable names, unquoted variable names, or tidyselect select helper functions. Default is `everything()`.
`show_single_row`	By default categorical variables are printed on multiple rows. If a variable is dichotomous (e.g. Yes/No) and you wish to print the regression coefficient on a single row, include the variable name(s) here–quoted and unquoted variable name accepted.
`conf.level`	Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`intercept`	Logical argument indicating whether to include the intercept in the output. Default is `FALSE`

estimate_fun	Function to round and format coefficient estimates. Default is style_sigfig when the coefficients are not transformed, and style_ratio when the coefficients have been exponentiated.
pvalue_fun	Function to round and format p-values. Default is style_pvalue. The function must have a numeric vector input (the numeric, exact p-value), and return a string that is the rounded/formatted p-value (e.g. `pvalue_fun = function(x) style_pvalue(x, digits = 2)` or equivalently, `purrr::partial(style_pvalue, digits = 2)`).
tidy_fun	Option to specify a particular tidier function if the model is not a vetted model or you need to implement a custom method. Default is NULL
show_yesno	DEPRECATED
exclude	DEPRECATED

Value

A tbl_regression object

Setting Defaults

If you prefer to consistently use a different function to format p-values or estimates, you can set options in the script or in the user- or project-level startup file, '.Rprofile'. The default confidence level can also be set.

- `options(gtsummary.pvalue_fun = new_function)`
- `options(gtsummary.tbl_regression.estimate_fun = new_function)`
- `options(gtsummary.conf.level = 0.90)`

Note

The N reported in the output is the number of observations in the data frame `model.frame(x)`. Depending on the model input, this N may represent different quantities. In most cases, it is the number of people or units in your model. Here are some common exceptions.

1. Survival regression models including time dependent covariates.
2. Random- or mixed-effects regression models with clustered data.
3. GEE regression models with clustered data.

This list is not exhaustive, and care should be taken for each number reported.

Example Output

Author(s)

Daniel D. Sjoberg

See Also

See tbl_regression vignette for detailed examples

Other tbl_regression tools: add_global_p.tbl_regression(), add_nevent.tbl_regression(), add_q(), bold_italicize_labels_levels, combine_terms(), inline_text.tbl_regression(), modify, tbl_merge(), tbl_stack()

Examples

```
# Example 1 ----------------------------------
library(survival)
tbl_regression_ex1 <-
  coxph(Surv(ttdeath, death) ~ age + marker, trial) %>%
  tbl_regression(exponentiate = TRUE)

# Example 2 ----------------------------------
tbl_regression_ex2 <-
  glm(response ~ age + grade, trial, family = binomial(link = "logit")) %>%
  tbl_regression(exponentiate = TRUE)

# Example 3 ----------------------------------
suppressMessages(library(lme4))
tbl_regression_ex3 <-
  glmer(am ~ hp + (1 | gear), mtcars, family = binomial) %>%
  tbl_regression(exponentiate = TRUE)
```

---

tbl_stack *Stacks two or more gtsummary objects*

Description

Assists in patching together more complex tables. `tbl_stack()` appends two or more `tbl_regression`, `tbl_summary`, `tbl_svysummary`, or `tbl_merge` objects. Column attributes, including number formatting and column footnotes, are retained from the first passed gtsummary object.

Usage

```
tbl_stack(tbls, group_header = NULL)
```

Arguments

`tbls`	List of gtsummary objects
`group_header`	Character vector with table headers where length matches the length of tbls=

Value

A `tbl_stack` object

Example Output

Author(s)

Daniel D. Sjoberg

See Also

tbl_merge

Other tbl_summary tools: add_n.tbl_summary(), add_overall(), add_p.tbl_summary(), add_q(), add_stat_label(), bold_italicize_labels_levels, inline_text.tbl_summary(), inline_text.tbl_survfit(), modify, tbl_merge(), tbl_summary()

Other tbl_svysummary tools: add_n.tbl_summary(), add_overall(), add_p.tbl_svysummary(), add_q(), add_stat_label(), modify, tbl_merge(), tbl_svysummary()

Other tbl_regression tools: add_global_p.tbl_regression(), add_nevent.tbl_regression(), add_q(), bold_italicize_labels_levels, combine_terms(), inline_text.tbl_regression(), modify, tbl_merge(), tbl_regression()

Other tbl_uvregression tools: add_global_p.tbl_uvregression(), add_nevent.tbl_uvregression(), add_q(), bold_italicize_labels_levels, inline_text.tbl_uvregression(), modify, tbl_merge(), tbl_uvregression()

Other tbl_survfit tools: add_n.tbl_survfit(), add_nevent.tbl_survfit(), add_p.tbl_survfit(), modify, tbl_merge(), tbl_survfit()

Examples

```
# Example 1 ----------------------------------
# stacking two tbl_regression objects
t1 <-
  glm(response ~ trt, trial, family = binomial) %>%
  tbl_regression(
    exponentiate = TRUE,
    label = list(trt ~ "Treatment (unadjusted)")
  )

t2 <-
  glm(response ~ trt + grade + stage + marker, trial, family = binomial) %>%
  tbl_regression(
    include = "trt",
    exponentiate = TRUE,
    label = list(trt ~ "Treatment (adjusted)")
  )

tbl_stack_ex1 <- tbl_stack(list(t1, t2))

# Example 2 ----------------------------------
# stacking two tbl_merge objects
library(survival)
t3 <-
  coxph(Surv(ttdeath, death) ~ trt, trial) %>%
  tbl_regression(
    exponentiate = TRUE,
    label = list(trt ~ "Treatment (unadjusted)")
  )

t4 <-
  coxph(Surv(ttdeath, death) ~ trt + grade + stage + marker, trial) %>%
  tbl_regression(
    include = "trt",
    exponentiate = TRUE,
    label = list(trt ~ "Treatment (adjusted)")
  )
```

```
# first merging, then stacking
row1 <- tbl_merge(list(t1, t3), tab_spanner = c("Tumor Response", "Death"))
row2 <- tbl_merge(list(t2, t4))
tbl_stack_ex2 <-
 tbl_stack(list(row1, row2), group_header = c("Unadjusted Analysis", "Adjusted Analysis"))
```

`tbl_summary` *Create a table of summary statistics*

Description

The `tbl_summary` function calculates descriptive statistics for continuous, categorical, and dichotomous variables. Review the tbl_summary vignette for detailed examples.

Usage

```
tbl_summary(
  data,
  by = NULL,
  label = NULL,
  statistic = NULL,
  digits = NULL,
  type = NULL,
  value = NULL,
  missing = NULL,
  missing_text = NULL,
  sort = NULL,
  percent = NULL,
  include = everything(),
  group = NULL
)
```

Arguments

`data`	A data frame
`by`	A column name (quoted or unquoted) in `data`. Summary statistics will be calculated separately for each level of the `by` variable (e.g. `by = trt`). If `NULL`, summary statistics are calculated using all observations.
`label`	List of formulas specifying variables labels, e.g. `list(age ~ "Age", stage ~ "Path T Stage")`. If a variable's label is not specified here, the label attribute (`attr(data$age,"label")`) is used. If attribute label is `NULL`, the variable name will be used.
`statistic`	List of formulas specifying types of summary statistics to display for each variable. The default is `list(all_continuous() ~ "{median} ({p25},{p75})",all_categorical() ~ "{n} ({p}%)")`. See below for details.
`digits`	List of formulas specifying the number of decimal places to round continuous summary statistics. If not specified, `tbl_summary` guesses an appropriate number of decimals to round statistics. When multiple statistics are displayed for a

	single variable, supply a vector rather than an integer. For example, if the statistic being calculated is `"{mean} ({sd})"` and you want the mean rounded to 1 decimal place, and the SD to 2 use `digits = list(age ~ c(1,2))`.
`type`	List of formulas specifying variable types. Accepted values are `c("continuous","categorical","` e.g. `type = list(age ~ "continuous",female ~ "dichotomous")`. If type not specified for a variable, the function will default to an appropriate summary type. See below for details.
`value`	List of formulas specifying the value to display for dichotomous variables. See below for details.
`missing`	Indicates whether to include counts of `NA` values in the table. Allowed values are `"no"` (never display NA values), `"ifany"` (only display if any NA values), and `"always"` (includes NA count row for all variables). Default is `"ifany"`.
`missing_text`	String to display for count of missing observations. Default is `"Unknown"`.
`sort`	List of formulas specifying the type of sorting to perform for categorical data. Options are `frequency` where results are sorted in descending order of frequency and `alphanumeric`, e.g. `sort = list(everything() ~ "frequency")`
`percent`	Indicates the type of percentage to return. Must be one of `"column"`, `"row"`, or `"cell"`. Default is `"column"`.
`include`	variables to include in the summary table. Default is `everything()`
`group`	DEPRECATED. Migrated to add_p

Value

A `tbl_summary` object

select helpers

Select helpers from the \tidyselect\ package and \gtsummary\ package are available to modify default behavior for groups of variables. For example, by default continuous variables are reported with the median and IQR. To change all continuous variables to mean and standard deviation use `statistic = list(all_continuous() ~ "{mean} ({sd})")`.

All columns with class logical are displayed as dichotomous variables showing the proportion of events that are `TRUE` on a single row. To show both rows (i.e. a row for `TRUE` and a row for `FALSE`) use `type = list(all_logical() ~ "categorical")`.

The select helpers are available for use in any argument that accepts a list of formulas (e.g. `statistic`, `type`, `digits`, `value`, `sort`, etc.)

statistic argument

The statistic argument specifies the statistics presented in the table. The input is a list of formulas that specify the statistics to report. For example, `statistic = list(age ~ "{mean} ({sd})")` would report the mean and standard deviation for age; `statistic = list(all_continuous() ~ "{mean} ({sd})")` would report the mean and standard deviation for all continuous variables. A statistic name that appears between curly brackets will be replaced with the numeric statistic (see glue::glue).

For categorical variables the following statistics are available to display.

- `{n}` frequency
- `{N}` denominator, or cohort size
- `{p}` formatted percentage

For continuous variables the following statistics are available to display.

- {median} median
- {mean} mean
- {sd} standard deviation
- {var} variance
- {min} minimum
- {max} maximum
- {p##} any integer percentile, where ## is an integer from 0 to 100
- {foo} any function of the form foo(x) is accepted where x is a numeric vector

For both categorical and continuous variables, statistics on the number of missing and non-missing observations and their proportions are available to display.

- {N_obs} total number of observations
- {N_miss} number of missing observations
- {N_nonmiss} number of non-missing observations
- {p_miss} percentage of observations missing
- {p_nonmiss} percentage of observations not missing

Note that for categorical variables, {N_obs}, {N_miss} and {N_nonmiss} refer to the total number, number missing and number non missing observations in the denominator, not at each level of the categorical variable.

type argument

tbl_summary displays summary statistics for three types of data: continuous, categorical, and dichotomous. If the type is not specified, tbl_summary will do its best to guess the type. Dichotomous variables are categorical variables that are displayed on a single row in the output table, rather than one row per level of the variable. Variables coded as TRUE/FALSE, 0/1, or yes/no are assumed to be dichotomous, and the TRUE, 1, and yes rows are displayed. Otherwise, the value to display must be specified in the value argument, e.g. value = list(varname ~ "level to show")

Example Output

Author(s)

Daniel D. Sjoberg

See Also

See tbl_summary vignette for detailed tutorial

See table gallery for additional examples

Other tbl_summary tools: add_n.tbl_summary(), add_overall(), add_p.tbl_summary(), add_q(), add_stat_label(), bold_italicize_labels_levels, inline_text.tbl_summary(), inline_text.tbl_survfit(modify, tbl_merge(), tbl_stack()

Examples

```
# Example 1 ----------------------------------
tbl_summary_ex1 <-
  trial[c("age", "grade", "response")] %>%
  tbl_summary()

# Example 2 ----------------------------------
tbl_summary_ex2 <-
  trial[c("age", "grade", "response", "trt")] %>%
  tbl_summary(
    by = trt,
    label = list(age ~ "Patient Age"),
    statistic = list(all_continuous() ~ "{mean} ({sd})"),
    digits = list(age ~ c(0, 1))
  )

# Example 3 ----------------------------------
# for convenience, you can also pass named lists to any arguments
# that accept formulas (e.g label, digits, etc.)
tbl_summary_ex3 <-
  trial[c("age", "trt")] %>%
  tbl_summary(
    by = trt,
    label = list(age = "Patient Age")
  )
```

---

tbl_survfit *Creates table of survival probabilities*

Description

Experimental Function takes a `survfit` object as an argument, and provides a formatted summary table of the results

Usage

```
tbl_survfit(x, ...)

## S3 method for class 'survfit'
tbl_survfit(
  x,
  times = NULL,
  probs = NULL,
  statistic = NULL,
  label = NULL,
  label_header = NULL,
  estimate_fun = NULL,
  missing = "—",
  conf.level = 0.95,
  reverse = FALSE,
  quiet = NULL,
  failure = NULL,
```

```
  ...
)

## S3 method for class 'data.frame'
tbl_survfit(
  x,
  y,
  times = NULL,
  probs = NULL,
  statistic = NULL,
  label = NULL,
  label_header = NULL,
  estimate_fun = NULL,
  missing = "—",
  conf.level = 0.95,
  reverse = FALSE,
  failure = NULL,
  include = everything(),
  quiet = NULL,
  ...
)

## S3 method for class 'list'
tbl_survfit(
  x,
  times = NULL,
  probs = NULL,
  statistic = NULL,
  label = NULL,
  label_header = NULL,
  estimate_fun = NULL,
  missing = "—",
  conf.level = 0.95,
  reverse = FALSE,
  quiet = NULL,
  ...
)
```

Arguments

x	a survfit object, list of survfit objects, or a data frame. If a data frame is passed, a list of survfit objects is constructed using each variable as a stratifying variable.
...	Not used
times	numeric vector of times for which to return survival probabilities.
probs	numeric vector of probabilities with values in (0,1) specifying the survival quantiles to return
statistic	string defining the statistics to present in the table. Default is `"{estimate} ({conf.low},{conf.high})"`
label	List of formulas specifying variables labels, e.g. `list(age ~ "Age,yrs", stage ~ "Path T Stage")`, or a string for a single variable table.
label_header	string specifying column labels above statistics. Default is `"{prob} Percentile"` for survival percentiles, and `"Time {time}"` for n-year survival estimates

estimate_fun	function to format the Kaplan-Meier estimates. Default is style_percent for survival probabilities and style_sigfig for survival times
missing	text to fill when estimate is not estimable. Default is "--"
conf.level	Confidence level for confidence intervals. Default is 0.95
reverse	Flip the probability reported, i.e. 1 -estimate. Default is FALSE. Does not apply to survival quantile requests
quiet	Logical indicating whether to print messages in console. Default is FALSE
failure	DEPRECATED. Use reverse= instead.
y	outcome call, e.g. y = Surv(ttdeath,death)
include	Variable to include as stratifying variables.

Example Output

Author(s)

Daniel D. Sjoberg

See Also

Other tbl_survfit tools: add_n.tbl_survfit(), add_nevent.tbl_survfit(), add_p.tbl_survfit(), modify, tbl_merge(), tbl_stack()

Examples

```
library(survival)

# Example 1 ----------------------------------
# Pass single survfit() object
tbl_survfit_ex1 <- tbl_survfit(
  survfit(Surv(ttdeath, death) ~ trt, trial),
  times = c(12, 24),
  label_header = "**{time} Month**"
)

# Example 2 ----------------------------------
# Pass a data frame
tbl_survfit_ex2 <- tbl_survfit(
  trial,
  y = survival::Surv(ttdeath, death),
  include = c(trt, grade),
  probs = 0.5,
  label_header = "**Median Survival**"
)

# Example 3 ----------------------------------
# Pass a list of survfit() objects
tbl_survfit_ex3 <-
  list(survfit(Surv(ttdeath, death) ~ 1, trial),
       survfit(Surv(ttdeath, death) ~ trt, trial)) %>%
  tbl_survfit(times = c(12, 24))

# Example 4 Competing Events Example ---------
```

```
# adding a competing event for death (cancer vs other causes)
library(dplyr, warn.conflicts = FALSE, quietly = TRUE)
trial2 <- trial %>%
  mutate(
  death_cr = case_when(
    death == 0 ~ "censor",
    runif(n()) < 0.5 ~ "death from cancer",
    TRUE ~ "death other causes"
  ) %>% factor()
)

survfit_cr_ex4 <-
  survfit(Surv(ttdeath, death_cr) ~ grade, data = trial2) %>%
  tbl_survfit(times = c(12, 24), label = "Tumor Grade")
```

`tbl_svysummary` *Create a table of summary statistics from a survey object*

Description

Experimental

Usage

```
tbl_svysummary(
  data,
  by = NULL,
  label = NULL,
  statistic = NULL,
  digits = NULL,
  type = NULL,
  value = NULL,
  missing = NULL,
  missing_text = NULL,
  sort = NULL,
  percent = NULL,
  include = NULL
)
```

Arguments

`data`	A survey object created with created with `survey::svydesign()`
`by`	A column name (quoted or unquoted) in `data`. Summary statistics will be calculated separately for each level of the `by` variable (e.g. `by = trt`). If `NULL`, summary statistics are calculated using all observations.
`label`	List of formulas specifying variables labels, e.g. `list(age ~ "Age", stage ~ "Path T Stage")`. If a variable's label is not specified here, the label attribute (`attr(data$age,"label")`) is used. If attribute label is `NULL`, the variable name will be used.
`statistic`	List of formulas specifying types of summary statistics to display for each variable. The default is `list(all_continuous() ~ "{median} ({p25},{p75})",all_categorical() ~ "{n} ({p}%)")`. See below for details.

digits	List of formulas specifying the number of decimal places to round continuous summary statistics. If not specified, tbl_summary guesses an appropriate number of decimals to round statistics. When multiple statistics are displayed for a single variable, supply a vector rather than an integer. For example, if the statistic being calculated is "{mean} ({sd})" and you want the mean rounded to 1 decimal place, and the SD to 2 use digits = list(age ~ c(1,2)).
type	List of formulas specifying variable types. Accepted values are c("continuous","categorical","d e.g. type = list(age ~ "continuous",female ~ "dichotomous"). If type not specified for a variable, the function will default to an appropriate summary type. See below for details.
value	List of formulas specifying the value to display for dichotomous variables. See below for details.
missing	Indicates whether to include counts of NA values in the table. Allowed values are "no" (never display NA values), "ifany" (only display if any NA values), and "always" (includes NA count row for all variables). Default is "ifany".
missing_text	String to display for count of missing observations. Default is "Unknown".
sort	List of formulas specifying the type of sorting to perform for categorical data. Options are frequency where results are sorted in descending order of frequency and alphanumeric, e.g. sort = list(everything() ~ "frequency")
percent	Indicates the type of percentage to return. Must be one of "column", "row", or "cell". Default is "column".
include	variables to include in the summary table. Default is everything()

Details

The tbl_svysummary function calculates descriptive statistics for continuous, categorical, and dichotomous variables taking into account survey weights and design. It is similar to tbl_summary().

Value

A tbl_svysummary object

statistic argument

The statistic argument specifies the statistics presented in the table. The input is a list of formulas that specify the statistics to report. For example, statistic = list(age ~ "{mean} ({sd})") would report the mean and standard deviation for age; statistic = list(all_continuous() ~ "{mean} ({sd})") would report the mean and standard deviation for all continuous variables. A statistic name that appears between curly brackets will be replaced with the numeric statistic (see glue::glue).

For categorical variables the following statistics are available to display.

- {n} frequency
- {N} denominator, or cohort size
- {p} formatted percentage
- {n_unweighted} unweighted frequency
- {N_unweighted} unweighted denominator
- {p_unweighted} unweighted formatted percentage

For continuous variables the following statistics are available to display.

- {median} median
- {mean} mean
- {sd} standard deviation
- {var} variance
- {min} minimum
- {max} maximum
- {p##} any integer percentile, where ## is an integer from 0 to 100
- {sum} sum

Unlike tbl_summary(), it is not possible to pass a custom function.

For both categorical and continuous variables, statistics on the number of missing and non-missing observations and their proportions are available to display.

- {N_obs} total number of observations
- {N_miss} number of missing observations
- {N_nonmiss} number of non-missing observations
- {p_miss} percentage of observations missing
- {p_nonmiss} percentage of observations not missing
- {N_obs_unweighted} unweighted total number of observations
- {N_miss_unweighted} unweighted number of missing observations
- {N_nonmiss_unweighted} unweighted number of non-missing observations
- {p_miss_unweighted} unweighted percentage of observations missing
- {p_nonmiss_unweighted} unweighted percentage of observations not missing

Note that for categorical variables, {N_obs}, {N_miss} and {N_nonmiss} refer to the total number, number missing and number non missing observations in the denominator, not at each level of the categorical variable.

Example Output

type argument

tbl_summary displays summary statistics for three types of data: continuous, categorical, and dichotomous. If the type is not specified, tbl_summary will do its best to guess the type. Dichotomous variables are categorical variables that are displayed on a single row in the output table, rather than one row per level of the variable. Variables coded as TRUE/FALSE, 0/1, or yes/no are assumed to be dichotomous, and the TRUE, 1, and yes rows are displayed. Otherwise, the value to display must be specified in the value argument, e.g. value = list(varname ~ "level to show")

select helpers

Select helpers from the \tidyselect\ package and \gtsummary\ package are available to modify default behavior for groups of variables. For example, by default continuous variables are reported with the median and IQR. To change all continuous variables to mean and standard deviation use statistic = list(all_continuous() ~ "{mean} ({sd})").

All columns with class logical are displayed as dichotomous variables showing the proportion of events that are TRUE on a single row. To show both rows (i.e. a row for TRUE and a row for FALSE) use type = list(all_logical() ~ "categorical").

The select helpers are available for use in any argument that accepts a list of formulas (e.g. statistic, type, digits, value, sort, etc.)

Author(s)

Joseph Larmarange

See Also

Other tbl_svysummary tools: add_n.tbl_summary(), add_overall(), add_p.tbl_svysummary(), add_q(), add_stat_label(), modify, tbl_merge(), tbl_stack()

Examples

```
# Example 1 ----------------------------------
# A simple weighted dataset
tbl_svysummary_ex1 <-
  survey::svydesign(~1, data = as.data.frame(Titanic), weights = ~Freq) %>%
  tbl_svysummary(by = Survived, percent = "row")

# Example 2 ----------------------------------
# A dataset with a complex design
data(api, package = "survey")
tbl_svysummary_ex2 <-
  survey::svydesign(id = ~dnum, weights = ~pw, data = apiclus1, fpc = ~fpc) %>%
  tbl_svysummary(by = "both", include = c(cname, api00, api99, both))
```

---

tbl_uvregression *Display univariate regression model results in table*

Description

This function estimates univariate regression models and returns them in a publication-ready table. It can create univariate regression models holding either a covariate or outcome constant.

For models holding outcome constant, the function takes as arguments a data frame, the type of regression model, and the outcome variable y=. Each column in the data frame is regressed on the specified outcome. The tbl_uvregression function arguments are similar to the tbl_regression arguments. Review the tbl_uvregression vignette for detailed examples.

You may alternatively hold a single covariate constant. For this, pass a data frame, the type of regression model, and a single covariate in the x= argument. Each column of the data frame will serve as the outcome in a univariate regression model. Take care using the x argument that each of the columns in the data frame are appropriate for the same type of model, e.g. they are all continuous variables appropriate for lm, or dichotomous variables appropriate for logistic regression with glm.

Usage

```
tbl_uvregression(
  data,
  method,
  y = NULL,
  x = NULL,
  method.args = NULL,
  exponentiate = FALSE,
  label = NULL,
  include = everything(),
  tidy_fun = NULL,
  hide_n = FALSE,
  show_single_row = NULL,
  conf.level = NULL,
  estimate_fun = NULL,
  pvalue_fun = NULL,
  formula = "{y} ~ {x}",
  show_yesno = NULL,
  exclude = NULL
)
```

Arguments

`data`	Data frame to be used in univariate regression modeling. Data frame includes the outcome variable(s) and the independent variables.
`method`	Regression method (e.g. lm, glm, survival::coxph, and more).
`y`	Model outcome (e.g. `y = recurrence` or `y = Surv(time,recur)`). All other column in `data` will be regressed on `y`. Specify one and only one of `y` or `x`
`x`	Model covariate (e.g. `x = trt`). All other columns in `data` will serve as the outcome in a regression model with `x` as a covariate. Output table is best when `x` is a continuous or dichotomous variable displayed on a single row. Specify one and only one of `y` or `x`
`method.args`	List of additional arguments passed on to the regression function defined by `method`.
`exponentiate`	Logical indicating whether to exponentiate the coefficient estimates. Default is `FALSE`.
`label`	List of formulas specifying variables labels, e.g. `list(age ~ "Age",stage ~ "Path T Stage")`
`include`	Variables to include in output. Input may be a vector of quoted variable names, unquoted variable names, or tidyselect select helper functions. Default is `everything()`.
`tidy_fun`	Option to specify a particular tidier function if the model is not a vetted model or you need to implement a custom method. Default is `NULL`
`hide_n`	Hide N column. Default is `FALSE`
`show_single_row`	By default categorical variables are printed on multiple rows. If a variable is dichotomous (e.g. Yes/No) and you wish to print the regression coefficient on a single row, include the variable name(s) here–quoted and unquoted variable name accepted.
`conf.level`	Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.

estimate_fun	Function to round and format coefficient estimates. Default is style_sigfig when the coefficients are not transformed, and style_ratio when the coefficients have been exponentiated.
pvalue_fun	Function to round and format p-values. Default is style_pvalue. The function must have a numeric vector input (the numeric, exact p-value), and return a string that is the rounded/formatted p-value (e.g. `pvalue_fun = function(x) style_pvalue(x, digits = 2)` or equivalently, `purrr::partial(style_pvalue, digits = 2)`).
formula	String of the model formula. Uses glue::glue syntax. Default is `"{y} ~ {x}"`, where `{y}` is the dependent variable, and `{x}` represents a single covariate. For a random intercept model, the formula may be `formula = "{y} ~ {x} + (1 | gear)"`.
show_yesno	DEPRECATED
exclude	DEPRECATED

Value

A `tbl_uvregression` object

Example Output

Setting Defaults

If you prefer to consistently use a different function to format p-values or estimates, you can set options in the script or in the user- or project-level startup file, '.Rprofile'. The default confidence level can also be set.

- `options(gtsummary.pvalue_fun = new_function)`
- `options(gtsummary.tbl_regression.estimate_fun = new_function)`
- `options(gtsummary.conf.level = 0.90)`

Note

The N reported in the output is the number of observations in the data frame `model.frame(x)`. Depending on the model input, this N may represent different quantities. In most cases, it is the number of people or units in your model. Here are some common exceptions.

1. Survival regression models including time dependent covariates.
2. Random- or mixed-effects regression models with clustered data.
3. GEE regression models with clustered data.

This list is not exhaustive, and care should be taken for each number reported.

Author(s)

Daniel D. Sjoberg

See Also

See tbl_regression vignette for detailed examples

Other tbl_uvregression tools: add_global_p.tbl_uvregression(), add_nevent.tbl_uvregression(), add_q(), bold_italicize_labels_levels, inline_text.tbl_uvregression(), modify, tbl_merge(), tbl_stack()

Examples

```
# Example 1 ----------------------------------
tbl_uv_ex1 <-
  tbl_uvregression(
    trial[c("response", "age", "grade")],
    method = glm,
    y = response,
    method.args = list(family = binomial),
    exponentiate = TRUE
  )

# Example 2 ----------------------------------
# rounding pvalues to 2 decimal places
library(survival)
tbl_uv_ex2 <-
  tbl_uvregression(
    trial[c("ttdeath", "death", "age", "grade", "response")],
    method = coxph,
    y = Surv(ttdeath, death),
    exponentiate = TRUE,
    pvalue_fun = function(x) style_pvalue(x, digits = 2)
  )
```

---

theme_gtsummary *Available gtsummary themes*

Description

Experimental The following themes are available to use within the gtsummary package. Use the set_gtsummary_theme() function to set a theme.

Usage

```
theme_gtsummary_journal(journal = c("jama", "lancet"), set_theme = TRUE)

theme_gtsummary_compact(set_theme = TRUE)

theme_gtsummary_printer(
 print_engine = c("gt", "kable", "kable_extra", "flextable", "huxtable", "tibble"),
  set_theme = TRUE
)

theme_gtsummary_language(
 language = c("de", "en", "es", "fr", "gu", "hi", "ja", "mr", "pt", "se", "zh-cn",
    "zh-tw"),
```

```
  decimal.mark = NULL,
  big.mark = NULL,
  iqr.sep = NULL,
  ci.sep = NULL,
  set_theme = TRUE
)
```

Arguments

journal	String indicating the journal theme to follow. • "jama" Journal of the American Medical Association
set_theme	Logical indicating whether to set the theme. Default is TRUE. When FALSE the named list of theme elements is returned invisibly.
print_engine	String indicating the print engine. Default is "gt"
language	String indicating language. Must be one of • "de" (German) • "en" (English) • "es" (Spanish) • "fr" (French) • "gu" (Gujarati) • "hi" (Hindi) • "ja" (Japanese) • "mr" (Marathi) • "pt" (Portuguese) • "se" (Swedish) • "zh-cn" Chinese (Simplified) • "zh-tw" Chinese (Traditional) If a language is missing a translation for a word or phrase, please feel free to reach out on GitHub with the translated text!
decimal.mark	The character to be used to indicate the numeric decimal point. Default is "." or getOption("OutDec")
big.mark	Character used between every 3 digits to separate hundreds/thousands/millions/etc. Default is ",", except when decimal.mark = "," when the default is a space.
iqr.sep	string indicating separator for the default IQR in tbl_summary(). If decimal.mark= is NULL, iqr.sep= is ",". The comma separator, however, can look odd when decimal.mark = ",". In this case the argument will default to an en dash
ci.sep	string indicating separator for confidence intervals. If decimal.mark= is NULL, ci.sep= is ",". The comma separator, however, can look odd when decimal.mark = ",". In this case the argument will default to an en dash

Themes

- theme_gtsummary_journal(journal=)
  - "jama"
    - sets theme to align with the JAMA reporting guidelines
    - large p-values are rounded to two decimal places

    - in `tbl_summary()` the IQR is separated with a dash, rather than comma
    - in `tbl_summary()` the percent symbol is not printed next to percentages
  - `"lancet"`
    - sets theme to align with the The Lancet reporting guidelines
    - large p-values are rounded to two decimal places
    - in `tbl_summary()` the IQR is separated with a dash, rather than comma
    - confidence intervals are separated with 4.5 to 7.8, rather than a comma
- `theme_gtsummary_compact()`
  - tables printed with gt or flextable will be compact with smaller font size and reduced cell padding
- `theme_gtsummary_printer(print_engine=)`
  - `"gt"` sets the gt package as the default print engine
  - `"kable"` sets the `knitr::kable()` function as the default print engine
  - `"flextable"` sets the flextable package as the default print engine
  - `"kable_extra"` sets the kableExtra package as the default print engine

Use `reset_gtsummary_theme()` to restore the default settings

Review the themes vignette to create your own themes.

Example Output

See Also

`set_gtsummary_theme()`

Themes vignette

`set_gtsummary_theme()`, `reset_gtsummary_theme()`

Examples

```
# Setting JAMA theme for gtsummary
theme_gtsummary_journal("jama")
# Themes can be combined by including more than one
theme_gtsummary_compact()

set_gtsummary_theme_ex1 <-
  trial %>%
  dplyr::select(age, grade, trt) %>%
  tbl_summary(by = trt) %>%
  add_stat_label() %>%
  as_gt()

# reset gtsummary themes
reset_gtsummary_theme()
```

`trial` *Results from a simulated study of two chemotherapy agents*

Description

A dataset containing the baseline characteristics of 200 patients who received Drug A or Drug B. Dataset also contains the outcome of tumor response to the treatment.

Usage

```
trial
```

Format

A data frame with 200 rows–one row per patient

trt Chemotherapy Treatment

age Age

marker Marker Level (ng/mL)

stage T Stage

grade Grade

response Tumor Response

death Patient Died

ttdeath Months to Death/Censor

Index